The component-by-component construction in weighted reproducing kernel Hilbert spaces – an optimization approach

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Mathematik

eingereicht von:	Adrian Ebert
geboren am:	07.10.1990
in:	Beeskow
Gutachter:	Prof. Andreas Griewank
	Dr. Hernan Leövey
eingereicht am:	2. November 2015

Acknowledgement

I would like to thank Prof. Andreas Griewank and Dr. Hernan Leövey for their excellent supervision in the course of this thesis. I do particularly appreciate the various occasions either of them took time for me and my issues and problems regarding the thesis. I want to especially thank Hernan Leövey for introducing me to the QMC theory as well as the associated research community. During the course of the thesis he has given me valuable suggestions and advice, based on which I developed the ideas featuring in this work.

I also want to thank Benjamin Drost and Robert Schleifer for their proof-reading of my thesis, which hopefully erased a large number of typos and mistakes.

Table of contents

1 Introduction

The goal of numerical integration is the approximation of some definite integral of a function f over a specific domain. Due to the development of high-powered computers in the past and recent century, the applicability of numerical integration methods could be expanded to high-dimensional integration problems in which the number of dimensions is in the hundreds and thousands. However, it is not possible to simply transfer one-dimensional methods to higher dimensions by separately applying them to every single coordinate direction. In contrast to one-dimensional integration methods, high-dimensional problems suffer from the so-called "curse of dimensionality", which means that the number of required coefficients for product integration rules grows exponentially in the dimension d. Instead, one approaches the problem to approximate a d-dimensional integral of the form

$$I(f) = \int_{[0,1]^d} f(x)\, dx$$

by so-called Quasi-Monte Carlo (QMC) methods. Quasi-Monte Carlo methods are equal-weight quadrature rules of the form

$$Q_{n,d}(f) = \frac{1}{n} \sum_{i=0}^{n-1} f(x_i)\,,$$

where the sample points $x_0, \dots, x_{n-1} \in [0,1]^d$ are chosen deterministically. The name Quasi-Monte Carlo originates from the classic Monte-Carlo methods which are similar to QMC methods with the only difference being that the quadrature points are chosen randomly from a uniform distribution on the unit cube $[0,1]^d$. The crucial point in the design of good QMC rules is the right choice of the underlying point set $P = \{x_0, \dots, x_{n-1}\}$. While the root-mean-square error of the Monte-Carlo method is

$$\mathbb{E}\left[(I(f) - Q_{n,d}(f))^2\right]^{\frac{1}{2}} = \frac{\sigma}{\sqrt{n}}\,,$$

and so the convergence rate of the MC method is of order $\mathcal{O}\left(1/\sqrt{n}\right)$, the Quasi-Monte Carlo approach has a considerably better convergence rate. It has been shown that for some low-discrepancy point sequences $\{x_i\}_{i\in\mathbb{N}}$ QMC methods can achieve a convergence rate that is of order $\mathcal{O}\left(\log(n)^d/n\right)$. Thus, it becomes apparent that QMC methods are asymptotically better than the original Monte Carlo quadrature rules, provided that the functions at hand satisfy suitable smoothness conditions. In particular, we will consider so-called lattice QMC rules which are quadrature rules with sample points belonging to some d-dimensional grid.

In the 1990s, the applicability of Quasi-Monte Carlo methods was extended to even higher dimensions, when Paskov and Traub (see [6] and [7]) showed that certain high-dimensional integration problems arising from finance could also be approached by means of QMC theory. In fact, they showed that for the respective integration problems of $d = 360$ dimensions the used QMC rules were superior to classic Monte-Carlo methods, which were at that time believed to be the best approach to such high-dimensional problems. Later, Sloan and Woźniakowski introduced in [10] and other works the concept of weighted reproducing kernel Hilbert spaces. In this class of function spaces the contribution of the different coordinates to the value of a particular d-dimensional function is quantified by an associated weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$. In [10] and [11], Sloan and Woźniakowski could prove that under certain conditions on the weight sequence γ numerical integration via QMC methods becomes tractable in the respective spaces.

Much research has been done in the theory of weighted reproducing kernel Hilbert spaces (WRKHS) since then, resulting in the design of a new method to construct point sets for QMC integration in WRKHS such that the integration remains tractable. This so-called component-by-component (CBC) construction finds generating vectors for rank-one lattice rules by minimizing a certain worst-case error expression one component at a time. The CBC construction is in particular interesting because it was proven by Kuo in [2] that the method achieves optimal rates of convergence. The component-by-component construction was then majorly improved by Nuyens and Cool (see [5]) who came up with a fast version of the original algorithm which reduced the time complexity of the method from $\mathcal{O}(d\,n^2)$ to only $\mathcal{O}(d\,n\log(n))$. This improvement broadened the importance and applicability of the CBC construction since it was now also possible to compute generating vectors for large values of n and dimensions d.

In general, the goal of the component-by-component construction is to find a generating vector z such that the corresponding worst-case error $e_{n,d}(z)$ is as small as possible. Even though the CBC algorithm works quite well to construct good generating vectors, the method does not address the question how to find the best possible generating vector z^* which minimizes $e_{n,d}(z)$ over the set of all possible generating vectors Z_n^d with $|Z_n^d| = (n-1)^d$. From an optimization point of view the CBC approach is a rather naive way to find this optimal vector as the algorithm searches coordinate-wise for the components of the generating vector. In the following thesis, we will therefore approach the search for good generating vectors for rank-one lattice rules in weighted RKHS from a different angle. At first we are going to regard the worst-case error expression as a d-dimensional, real-valued function and apply standard optimization methods to it in order to solve the related minimization problem. Moreover, we will implement and study a new kind of algorithm which can be regarded as a generalized version of the standard component-by-component construction.

2 Quasi-Monte Carlo rules

2.1 Integration rules

As mentioned in the introduction, the goal of numerical integration is to approximate the multivariate integral of a function f by a quadrature rule, i.e.

$$\int_{[0,1]^d} f(x)\,dx \approx \sum_{i=0}^{n-1} w_i f(x_i)\,,$$

where $P = \{x_0, \dots, x_{n-1}\} \subseteq [0,1]^d$ is the point set of quadrature points and the quadrature weights $w_0, \dots, w_{n-1} \in [0,1]$ are such that $\sum_{i=0}^{n-1} w_i = 1$. This last condition assures that the quadrature rule is exact for constant functions.

General Assumptions: In this thesis we make the following assumptions:

(1) The domain of integration is the d-dimensional unit cube $[0,1]^d$. This is a reasonable assumption since most integration problems in practice can be transformed to the unit cube.

(2) The considered functions f belong to some normed function space H, are integrable and have a certain level of smoothness.

(3) We consider only equal-weight quadrature rules, i.e. $w_i = \frac{1}{n}$ for all $i = 1, \dots, n$.

Based on these assumptions we introduce a special class of quadrature rules. These so-called Quasi-Monte Carlo rules will be the centre of this thesis and are defined as follows.

Definition 2.1 (Quasi-Monte Carlo method)
A Quasi-Monte Carlo (QMC) method is a quadrature rule of the form

$$Q_{n,d}(f) = \frac{1}{n} \sum_{i=0}^{n-1} f(x_i)\,,$$

where the quadrature points $x_0, \dots, x_{n-1} \in [0,1]^d$ are chosen deterministically.

Having constructed a particular QMC rule $Q_{n,d}$, it is natural to ask how accurate the integration rule approximates the integral $\int_{[0,1]^d} f(x)\,dx$. We are therefore interested in the integration error

$$e_{n,d}(Q_{n,d}, f) = \int_{[0,1]^d} f(x)\,dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) = I(f) - Q_{n,d}(f)\,,$$

where $f : [0,1]^d \to \mathbb{R}$ is some function in H. However, this integration error only gives us information about the accuracy of approximation for a particular function f. To make statements about the quality of a certain Quasi-Monte Carlo rule $Q_{n,d}$ for the entire function space H, we introduce a different error notion, the so-called worst-case error.

Definition 2.2 (Worst-case error)
Let $Q_{n,d}$ be a Quasi-Monte Carlo rule in the normed function space H with underlying point-set $P = \{x_0, \dots, x_{n-1}\} \subseteq [0,1]^d$. The worst-case error of $Q_{n,d}$ is then defined as

$$e_{n,d}(Q_{n,d}, H) = \sup_{\|f\|_H \leq 1} \left| \int_{[0,1]^d} f(x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| = \sup_{\|f\|_H \leq 1} |I(f) - Q_{n,d}(f)| \,.$$

In other words $e_{n,d}(Q_{n,d}, H)$ is the worst error that is attained by a function in the unit ball of H.

Remark: This definition is particularly interesting because of the following relation:
Let f be any function in H. Due to the linearity of $Q_{n,d}$ and the integral we obtain that

$$\begin{aligned}
|I(f) - Q_{n,d}(f)| &= \left| \int_{[0,1]^d} f(x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| \\
&= \left| \int_{[0,1]^d} \|f\| \cdot \frac{f(x)}{\|f\|}\, dx - \frac{1}{n} \sum_{i=0}^{n-1} \|f\| \cdot \frac{f(x_i)}{\|f\|} \right| \\
&= \|f\| \cdot \left| \int_{[0,1]^d} \frac{f(x)}{\|f\|}\, dx - \frac{1}{n} \sum_{i=0}^{n-1} \frac{f(x_i)}{\|f\|} \right| \\
&= \|f\| \cdot \left| \int_{[0,1]^d} g(x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} g(x_i) \right| \text{, where } g := \frac{f}{\|f\|} \\
&\leq \|f\| \cdot e_{n.d}(Q_{n,d}, H) \,.
\end{aligned}$$

The last step follows since $\|g\|_H = \left\| \frac{f}{\|f\|} \right\| = \frac{\|f\|}{\|f\|} = 1$ and so by definition of the worst-case error

$$\left| \int_{[0,1]^d} g(x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} g(x_i) \right| \leq \sup_{\|f\|_H \leq 1} \left| \int_{[0,1]^d} f(x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| = e_{n,d}(Q_{n,d}, H) \,.$$

Thus, we have for any function $f \in H$ that

$$|I(f) - Q_{n,d}(f)| \leq e_{n.d}(Q_{n,d}, H) \cdot \|f\|_H \,.$$

Furthermore, we introduce the initial error $e_{0,d}(H)$ which will be used in the subsequent chapters.

$$e_{0,d}(H) := \sup_{\|f\|_H \leq 1} \left| \int_{[0,1]^d} f(x)\, dx \right|$$

Even though the notion of the worst-case error allows us to evaluate the accuracy of a particular QMC rule in the used function space H, it is in general difficult to compute the worst-case error. In some spaces, however, it is possible to derive explicit expressions for the worst-case error. These so-called reproducing kernel Hilbert spaces will be introduced in the next section.

2.2 Reproducing kernel Hilbert spaces

Definition 2.3 (Reproducing kernel Hilbert space)
Let H be a Hilbert space of real-valued functions $f : [0,1]^d \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_H$. Then H is called a reproducing kernel Hilbert space if there exists a kernel $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$ such that

- *$K(\cdot, x) \in H$ for all $x \in [0,1]^d$*
- *$f(x) = \langle f, K(\cdot, x) \rangle_H$ for all $f \in H$ and for all $x \in [0,1]^d$.*

Remark: A kernel function $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$ with the above properties also satisfies:

1. $K(x,y) = K(y,x)$ for all $x, y \in [0,1]^d$ (symmetry)
   By the reproducing property of the kernel function K we have for any $x, y \in [0,1]^d$

$$K(x,y) = \langle K(\cdot, y), K(\cdot, x) \rangle = \langle K(\cdot, x), K(\cdot, y) \rangle = K(y,x)\,.$$

2. The kernel function K of a reproducing kernel Hilbert space is unique. To see this, assume there are two kernels K and $\widetilde{K}$ such that $\langle f, K(\cdot, x) \rangle_H = \langle f, \widetilde{K}(\cdot, x) \rangle_H$ for all $f \in H$ and for all $x \in [0,1]^d$. Then we have for every $x \in [0,1]^d$ that

$$0 = \langle f, K(\cdot, x) - \widetilde{K}(\cdot, x) \rangle_H \text{ for all } f \in H\,.$$

   Since H is a Hilbert space, this implies that $K(\cdot, x) - \widetilde{K}(\cdot, x) = 0 \Leftrightarrow K(\cdot, x) = \widetilde{K}(\cdot, x)$ for all $x \in [0,1]^d$ and thus $K = \widetilde{K}$. Therefore the kernel function of a RKHS is unique.

The definition naturally raises the question when a particular Hilbert space of real-valued functions possesses a reproducing kernel. To answer this question we derive an equivalent condition for the existence of a kernel function with the above properties.

Theorem 2.4
Let H be a Hilbert space of functions $f : [0,1]^d \to \mathbb{R}$. Then H is a reproducing kernel Hilbert space if and only if the point evaluation functionals $F_x : H \to \mathbb{R}$ are continuous for every $x \in [0,1]^d$, where F_x is given by $F_x(f) = f(x)$.

Proof. Assume H is a reproducing kernel Hilbert space with kernel K and consider an arbitrary $x \in [0,1]^d$. Using the reproducing property, we obtain for $f \in H$ with $f \neq 0$ that

$$\begin{aligned} |F_x(f)| = |f(x)| = |\langle f, K(\cdot, x) \rangle| &\overset{\text{CS}}{\leq} \|f\| \cdot \|K(\cdot, x)\|\,. \\ \Rightarrow \frac{|F_x(f)|}{\|f\|} &\leq \|K(\cdot, x)\| \text{ for all } f \in H \text{ with } f \neq 0 \\ \Rightarrow \|F_x\| = \sup_{f \in H, f \neq 0} \frac{|F_x(f)|}{\|f\|} &\leq \|K(\cdot, x)\| \end{aligned}$$

Since $\|K(\cdot, x)\| = \sqrt{\langle K(\cdot, x), K(\cdot, x) \rangle} = \sqrt{K(x,x)} < \infty$, we have that $\|F_x\| \leq \sqrt{K(x,x)}$ and thus F_x is bounded, i.e. continuous. Hence F_x is a continuous functional for all $x \in [0,1]^d$.

Conversely, if we assume that the point evaluation functional F_x is continuous for every $x \in [0,1]^d$, the Riesz representation theorem implies that for all $x \in [0,1]^d$ there exists a unique $K_x \in H$ such that

$$f(x) = F_x(f) = \langle f, K_x \rangle_H \quad \forall f \in H\,.$$

Using this identity, we define the kernel function $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$ by $K(\cdot, x) = K_x \in H$. Then we have that $K(\cdot, x) \in H$ for all $x \in [0,1]^d$ and further that $f(x) = \langle f, K(\cdot, x)\rangle_H$ for all $f \in H$. Therefore, we have constructed the reproducing kernel K. Hence, if the point evaluation functionals in a Hilbert space are continuous there exists a reproducing kernel K. ■

In order to derive the desired explicit expression for the worst-case error, we prove the following Lemma which states that under certain conditions integration on $[0,1]^d$ and inner product taking are interchangeable.

Lemma 2.5
Let H be a reproducing kernel Hilbert space with kernel function $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$. Further, assume that the integration functional I is continuous on H, where $I(f)$ is given by

$$I(f) = \int_{[0,1]^d} f(x)\,dx\,.$$

Then we have for all $f \in H$ that

$$\int_{[0,1]^d} \langle f, K(\cdot, y)\rangle_H \,dy = \left\langle f, \int_{[0,1]^d} K(\cdot, y)\,dy \right\rangle_H .$$

Proof. Since by assumption I is a continuous linear functional, the Riesz representation theorem implies that there exists a function $\xi \in H$ such that $I(f) = \int_{[0,1]^d} f(x)\,dx = \langle f, \xi\rangle$ for all $f \in H$. Due to the reproducing kernel property, we obtain for arbitrary $x \in [0,1]^d$ that

$$\xi(x) = \langle \xi, K(\cdot, x)\rangle = \langle K(x, \cdot), \xi\rangle = I(K(x, \cdot)) = \int_{[0,1]^d} K(x, y)\,dy\,.$$

The original claim now easily follows since

$$\int_{[0,1]^d} \langle f, K(\cdot, y)\rangle_H \,dy = \int_{[0,1]^d} f(y)\,dy = I(f) = \langle f, \xi\rangle = \left\langle f, \int_{[0,1]^d} K(\cdot, y)\,dy \right\rangle_H .$$

■

Remark: The continuity of the integration functional I is rather natural. Novak and Woźniakowski showed in [4, Section 23.4] that in a Hilbert space of integrable functions the continuity of the point evaluation functionals F_x is sufficient for that property. We can further derive the following sufficient condition for the continuity of the integration functional. Recall that we found in the proof of 2.4 that $|F_x(f)| \leq \sqrt{K(x,x)}$. Hence, we obtain for any $f \in H$ with $f \neq 0$

$$\begin{aligned} |I(f)| &= \left| \int_{[0,1]^d} f(x)\,dx \right| = \left| \int_{[0,1]^d} F_x(f)\,dx \right| \\ &\leq \int_{[0,1]^d} |F_x(f)|\,dx \leq \int_{[0,1]^d} \sqrt{K(x,x)}\,dx\,. \end{aligned}$$

Thus, if the kernel function K satisfies $\int_{[0,1]^d} \sqrt{K(x,x)}\,dx < \infty$, then I is a continuous functional. In particular, the continuity of the integration functional I also implies that the error functional

$$e_{n,d}(Q_{n,d}, f) = \int_{[0,1]^d} f(x)\,dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$

is a continuous functional. To see this, note that the functional $Q_{n,d}(f)$ given by

$$Q_{n,d}(f) = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) = \frac{1}{n}\sum_{i=0}^{n-1} F_{x_i}(f)$$

is the sum of the point evaluation functionals F_{x_i} which are continuous provided H is a reproducing kernel Hilbert space. Hence, $Q_{n,d}$ is continuous and therefore $e_{n,d} = I - Q_{n,d}$ is also continuous.

As mentioned in the last section, we can now derive an explicit formula for the worst-case error.

Theorem 2.6 (Worst-case error expression)
Let H be a reproducing kernel Hilbert space with kernel function $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$ in which integration is continuous. Then the squared worst-case error of a Quasi-Monte Carlo rule $Q_{n,d}$ with quadrature points $\{x_0, \dots, x_{n-1}\}$ takes the following form

$$e^2_{n,d}(Q_{n,d}, H) = \int_{[0,1]^d}\int_{[0,1]^d} K(x,y)\,dx\,dy - \frac{2}{n}\sum_{i=0}^{n-1}\int_{[0,1]^d} K(x_i,x)\,dx + \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} K(x_i,x_j)\,.$$

Proof. Since by assumption I is continuous, we can apply Lemma 2.5. This together with the reproducing property of K gives the following two expressions

$$\int_{[0,1]^d} f(x)\,dx = \int_{[0,1]^d} \langle f, K(\cdot,x)\rangle\,dx \overset{2.5}{=} \left\langle f, \int_{[0,1]^d} K(\cdot,x)\,dx \right\rangle$$

$$\frac{1}{n}\sum_{i=0}^{n-1} f(x_i) = \frac{1}{n}\sum_{i=0}^{n-1}\langle f, K(\cdot,x_i)\rangle = \left\langle f, \frac{1}{n}\sum_{i=0}^{n-1} K(\cdot,x_i)\right\rangle\,.$$

Combining both expressions, we find for the integration error

$$\int_{[0,1]^d} f(x)\,dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) = \left\langle f, \underbrace{\int_{[0,1]^d} K(\cdot,x)\,dx - \frac{1}{n}\sum_{i=0}^{n-1} K(\cdot,x_i)}_{=:\zeta}\right\rangle = \langle f, \zeta\rangle$$

We see that ζ is the unique representer of the bounded linear functional $e_{n,d}$ whose existence is guaranteed by the Riesz representation theorem. For the worst-case error this gives

$$e_{n,d}(Q_{n,d}, H) = \sup_{\|f\|\le 1} |\langle f, \zeta\rangle| \overset{CS}{\le} \|f\|\cdot\|\zeta\| \quad \text{with} \quad \|f\| \le 1\,.$$

Choosing $f = \frac{\zeta}{\|\zeta\|}$, we obtain that $|\langle f, \zeta \rangle| = \frac{|\langle \zeta, \zeta \rangle|}{\|\zeta\|} = \|\zeta\|$ and thus our f attains the supremum, i.e. $e^2_{n,d}(Q_{n,d}, H) = \|\zeta\|^2 = \langle \zeta, \zeta \rangle$. Using the definition of ζ we obtain

$$\begin{aligned} e^2_{n,d}(Q_{n,d}, H) &= \langle \zeta, \zeta \rangle = \left\langle \int_{[0,1]^d} K(\cdot, x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} K(\cdot, x_i), \int_{[0,1]^d} K(\cdot, x)\, dx - \frac{1}{n} \sum_{i=0}^{n-1} K(\cdot, x_i) \right\rangle \\ &= \left\langle \int_{[0,1]^d} K(\cdot, x)\, dx, \int_{[0,1]^d} K(\cdot, y)\, dy \right\rangle - \frac{2}{n} \sum_{i=0}^{n-1} \left\langle \int_{[0,1]^d} K(\cdot, x)\, dx, K(\cdot, x_i) \right\rangle + \\ &\quad + \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \langle K(\cdot, x_j), K(\cdot, x_i) \rangle \\ &= \int_{[0,1]^d} \int_{[0,1]^d} K(x, y)\, dx\, dy - \frac{2}{n} \sum_{i=0}^{n-1} \int_{[0,1]^d} K(x_i, x)\, dx + \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} K(x_i, x_j)\,. \end{aligned}$$

Here, we repeatedly used the interchangeability of the integral and the inner product as well as the reproducing kernel property. ■

3 Weighted spaces

3.1 The idea behind weighted spaces

In general, numerical integration problems are particularly difficult if the nominal dimension of the problem is large. However, there are certain types of problems which are much easier to solve than the actual dimension would indicate. In 1995, Paskov and Traub presented a class of integration problems called collateralized mortgage obligations (CMO) from the field of financial derivatives (compare to [6] and [7]). In short, these CMO's are packages of mortgages held by a bank or financial institution. A bank is naturally interested in the value of its CMO's which is affected by the fact whether or not a monthly mortgage loan is repaid by the obligor. Since the value of a mortgage package is determined by the monthly behaviour of the involved borrowers over the 30 year repayment period, the problem is of very high dimension. In mathematical terms the problem can be formulated as the calculation of a 360-dimensional expectation value. Paskov and Traub used a Quasi-Monte Carlo method to approach this high-dimensional integration problem and, surprisingly, came up with satisfactory results. In particular, the used QMC method was superior to the standard Monte-Carlo method, which was at that time believed to be the best way to tackle high-dimensional integration problems.

Due to the success of the QMC methods for certain classes of high-dimensional problems, Caflisch, Morokov and Owen introduced the notion of "effective dimension" (in superposition and truncation sense). The idea behind this definition is that the variables x_j of a function $f : \mathbb{R}^d \to \mathbb{R}$ have a varying importance regarding the impact on the function value. Thus, there exist functions with nominal dimension d which have an effective dimension that is much smaller than d. Consequently, the corresponding integration problems also inherit an effective dimension since the more important variables contribute stronger to the value of the integral than the others. Therefore, some integrals can be approximated quite accurately by means of QMC theory even though the nominal dimension is high.

To quantify the varying importance of the coordinates of a function, we arrange the variables in decreasing order with respect to their importance and then associate with every coordinate direction x_j a positive number γ_j which reflects the importance of x_j. Based on this idea, we introduce weighted function spaces with incorporated weights $\{\gamma_j\}_{j=1}^d$, where $\gamma_1 \geq \gamma_2 \geq \gamma_3 \geq \ldots > 0$. In the successive sections we will introduce two different weighted function spaces: the weighted Sobolev space and the weighted Korobov space. These two function spaces will be the main focus of our numerical experiments in the chapters 5, 6 and 7.

3.2 The weighted anchored Sobolev space

To construct the weighted Sobolev space, we return to the integration of functions $f : [0, 1] \to \mathbb{R}$ in one dimension, where f is continuously differentiable. For this kind of functions, the fundamental theorem of calculus implies

$$\int_x^1 f'(y)\, dy = f(1) - f(x)$$

and so we obtain as expression for $f(x)$

$$f(x) = f(1) - \int_x^1 f'(y)\,dy\,.$$

Using this identity, we can derive an expression for the integration error

$$e_{n,1}(Q_n, f) = \int_0^1 f(x)\,dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)$$

for a one-dimensional QMC rule Q_n with quadrature points $x_0, \ldots, x_{n-1} \in [0,1]$ as follows.

$$\begin{aligned}
e_{n,1}(Q_n, f) &= \int_0^1 f(x)\,dx - \frac{1}{n}\sum_{i=0}^{n-1} f(x_i) \\
&= \int_0^1 \left(f(1) - \int_x^1 f'(y)\,dy\right) dx + \frac{1}{n}\sum_{i=0}^{n-1}(f(1) - f(1) - f(x_i)) \\
&= f(1) - \int_0^1 \int_x^1 f'(y)\,dy\,dx - \frac{1}{n}\sum_{i=0}^{n-1} f(1) + \frac{1}{n}\sum_{i=0}^{n-1}(f(1) - f(x_i)) \\
&= \frac{1}{n}\sum_{i=0}^{n-1}\int_{x_i}^1 f'(y)\,dy - \int_0^1 \int_x^1 f'(y)\,dy\,dx \\
&= \int_0^1 \frac{1}{n}\sum_{i=0}^{n-1}(\mathbb{1}_{[x_i,1]}(y)\cdot f'(y))\,dy - \int_0^1 \underbrace{\int_0^y f'(y)\,dx}_{=f'(y)\cdot y}\,dy \\
&= \int_0^1 \left[\frac{1}{n}\sum_{i=0}^{n-1}(\mathbb{1}_{[x_i,1]}(y) - y)\right]\cdot f'(y)\,dy
\end{aligned}$$

Now note that due to the following two relations

(1) $y \in [x_i, 1] \iff x_i \le y \le 1 \iff 0 \le x_i \le y \iff x_i \in [0, y]$

(2) $y \notin [x_i, 1] \iff 0 \le y < x_i \iff y < x_i \le 1 \iff x_i \notin [0, y]$

we have that $\mathbb{1}_{[x_i,1]}(y) = \mathbb{1}_{[0,y]}(x_i)$ for all $y \in [0,1]$. Thus, the above expression simplifies to

$$e_{n,1}(Q_n, f) = \int_0^1 \left[\frac{1}{n}A([0,y], P_n) - \lambda([0,y])\right]\cdot f'(y)\,dy = \int_0^1 \Delta_{P_n}(y)\cdot f'(y)\,dy\,,$$

where P_n is the point set $P_n = \{x_0, \ldots, x_{n-1}\}$, $A(\cdot)$ is the standard counting function and $\Delta_{P_n}(y)$ denotes the local discrepancy of P_n at y given by

$$\Delta_{P_n}(y) := \frac{1}{n}A([0,y], P_n) - \lambda([0,y]) = \frac{1}{n}A([0,y], P_n) - y\,.$$

By applying Hölder's inequality, we obtain furthermore

$$\begin{aligned} |e_{n,1}(Q_n, f)| &\leq \int_0^1 |\Delta_{P_n}(y)| \cdot |f'(y)| \, dy \leq \left(\int_0^1 |f'(y)|^q \, dy \right)^{\frac{1}{q}} \cdot \left(\int_0^1 |\Delta_{P_n}(y)|^p \, dy \right)^{\frac{1}{p}} \\ &= \|f'\|_{L^q} \cdot \|\Delta_{P_n}\|_{L^p} = \|f'\|_{L^q} \cdot L_p(P_n) \,, \end{aligned}$$

where $L_p(P_n)$ denotes the L_p-discrepancy of the point set P_n and $p, q \in [1, \infty]$ are such that $\frac{1}{p} + \frac{1}{q} = 1$.

In order to construct a normed function space, we define the norm $\|f\|_{1,q}$ for continuously differentiable functions $f : [0,1] \to \mathbb{R}$ by

$$\|f\|_{1,q} := \left(|f(1)|^q + \int_0^1 |f'(y)|^q \, dy \right)^{\frac{1}{q}} .$$

We immediately see that $\|f\|_{1,q} \geq \|f'\|_{L^q}$ and hence obtain as an intermediate result

$$|e_{n,1}(Q_n, f)| \leq \|f\|_{1,q} \cdot L_p(P_n) \, .$$

Remark: For $p = \infty$ and $q = 1$ the above inequality becomes the familiar Koksma-Hlawka inequality given by:

$$|e_{n,1}(Q_n, f)| \leq \|f\|_{1,1} \cdot D_n^*(P_n) \, .$$

Note that $\|\Delta_{P_n}\|_{L^\infty} = \sup_{y \in [0,1]} |\Delta_{P_n}(y)| = \sup_{y \in [0,1]} \left| \frac{1}{n} A([0,y], P_n) - \lambda([0,y]) \right| =: D_n^*(P_n)$.

Turning back to our function space, we choose $q = 2$ and obtain as norm

$$\|f\|_{1,2} := \left(|f(1)|^2 + \int_0^1 |f'(y)|^2 \, dy \right)^{\frac{1}{2}} .$$

The resulting function space is given by

$$\mathcal{H}_1 := \{ f : [0,1] \to \mathbb{R} \,,\ f \text{ is absolutely continuous and } \|f\|_{1,2} < \infty \} \,.$$

Remember that we are in particular interested in reproducing kernel Hilbert spaces since by Theorem 2.6 these spaces give rise to an explicit error expression. Therefore, we equip the space $\mathcal{H}_1$ with an inner product defined by

$$\langle f, g \rangle := f(1) \cdot g(1) + \int_0^1 f'(x) \cdot g'(x) \, dx \,.$$

The induced norm $\| \cdot \| = \sqrt{\langle \cdot, \cdot \rangle}$ is exactly the norm $\|f\|_{1,2}$ we considered before. The desired reproducing kernel of the Hilbert space $\mathcal{H}_1$ is $K_1(x,y) = 2 - \max(x,y)$ with corresponding partial derivative with respect to x given by

$$\frac{\partial K_1(x,y)}{\partial x} = \begin{cases} -1 & \text{, if } x > y \,, \\ 0 & \text{, if } x \leq y \,. \end{cases}$$

We see that $K_1(\cdot, y)$ is absolutely continuous and furthermore integrable for all $y \in [0,1]$. Additionally, we have for all $y \in [0,1]$:

$$\begin{aligned} \|K_1(\cdot,y)\|_{1,2}^2 &= \langle K_1(\cdot,y), K_1(\cdot,y)\rangle = K_1(1,y)^2 + \int_0^1 \left(\frac{\partial K_1(x,y)}{\partial x}\right)^2 dx \\ &= (2-\max(1,y))^2 + \int_y^1 (-1)^2\, dx = 1 + \int_y^1 dx = 1+1-y = 2-y < \infty\,. \end{aligned}$$

Hence, we have that $K_1(\cdot, y) \in \mathcal{H}_1$ for all $y \in [0,1]$. We are left to check the reproducing property of the kernel function K_1:

$$\begin{aligned} \langle f, K_1(\cdot,y)\rangle &= f(1)\cdot K_1(1,y) + \int_0^1 f'(x)\cdot\left(\frac{\partial K_1(x,y)}{\partial x}\right) dx \\ &= f(1)\cdot(2-\max(1,y)) - \int_y^1 f'(x)\, dx = f(1) - f(1) + f(y) = f(y)\,. \end{aligned}$$

Thus, $K_1(x,y)$ is the uniquely defined reproducing kernel for the RKHS $\mathcal{H}_1$.

The Sobolev space $\mathcal{H}_1$ in one dimension which we constructed above can be modified to the weighted Sobolev space $\mathcal{H}_{1,\gamma}$ with incorporated weight $\gamma > 0$. Again $\mathcal{H}_{1,\gamma}$ contains all absolutely continuous functions $f : [0,1] \to \mathbb{R}$ whose first derivatives are square-integrable. But now we associate to $\mathcal{H}_{1,\gamma}$ a positive weight γ and the anchor value c, where $c \in [0,1]$. Based on that, we define the inner-product of $\mathcal{H}_{1,\gamma}$ by

$$\langle f, g\rangle_{1,\gamma} := f(c)\cdot g(c) + \frac{1}{\gamma}\int_0^1 f'(x)\cdot g'(x)\, dx\,,$$

and further set $\|f\|_{1,\gamma} = \sqrt{\langle f, f\rangle_{1,\gamma}}$. This space is again a reproducing kernel Hilbert space and the reproducing kernel is given by

$$K_{1,\gamma}(x,y) = 1 + \gamma\cdot\eta(x,y)\,,$$

where

$$\eta(x,y) := \begin{cases} \min(x,y) - c & \text{, if } x,y > c\,, \\ c - \max(x,y) & \text{, if } x,y < c\,, \\ 0 & \text{, otherwise}\,. \end{cases}$$

One can easily check that $K_{1,\gamma}$ satisfies the reproducing property for $\mathcal{H}_{1,\gamma}$:

Depending on c and y, we obtain for the partial derivative with respect to x:

(i) $\underline{y > c:}$

$$\frac{\partial K_{1,\gamma}(x,y)}{\partial x} = \begin{cases} 0 & \text{, if } x \geq y\,, \\ \gamma & \text{, if } c < x < y\,, \\ 0 & \text{, if } x \leq c\,. \end{cases}$$

(ii) $\underline{y < c:}$

$$\frac{\partial K_{1,\gamma}(x,y)}{\partial x} = \begin{cases} 0 & \text{, if } x < y\,, \\ -\gamma & \text{, if } y \le x < c\,, \\ 0 & \text{, if } x \ge c\,. \end{cases}$$

Using these two expressions, we obtain:

(i) $\underline{y > c:}$

$$\begin{aligned} \langle f, K_{1,\gamma}(\cdot,y)\rangle_{1,\gamma} &= f(c)\cdot K_{1,\gamma}(c,y) + \frac{1}{\gamma}\int_0^1 f'(x)\cdot\frac{\partial K_{1,\gamma}(x,y)}{\partial x}\,dx \\ &= f(c)\cdot(1+\gamma\cdot\underbrace{\eta(c,y)}_{=0}) + \frac{1}{\gamma}\int_c^y f'(x)\cdot\gamma\,dx = f(c) + \int_c^y f'(x)\,dx \\ &= f(c) + f(y) - f(c) = f(y) \end{aligned}$$

(ii) $\underline{y < c:}$

$$\begin{aligned} \langle f, K_{1,\gamma}(\cdot,y)\rangle_{1,\gamma} &= f(c)\cdot K_{1,\gamma}(c,y) + \frac{1}{\gamma}\int_0^1 f'(x)\cdot\frac{\partial K_{1,\gamma}(x,y)}{\partial x}\,dx \\ &= f(c) + \frac{1}{\gamma}\int_y^c (-\gamma)\cdot f'(x)\,dx = f(c) - \int_y^c f'(x)\,dx \\ &= f(c) - f(c) + f(y) = f(y) \end{aligned}$$

(iii) $\underline{y = c:}$ Then $K_{1,\gamma}(x,y) = 1$ for all $x \in [0,1]$ and thus

$$\langle f, K_{1,\gamma}(\cdot,y)\rangle_{1,\gamma} = f(c) + \frac{1}{\gamma}\int_0^1 f'(x)\cdot 0\,dx = f(c) = f(y)\,.$$

Hence, $K_{1,\gamma}$ satisfies the reproducing property and so $\mathcal{H}_{1,\gamma}$ is indeed a RKHS.

The reproducing kernel Hilbert space $\mathcal{H}_{1,\gamma}$ can now be used to construct a similar d-dimensional version of $\mathcal{H}_{1,\gamma}$. We define the space $\mathcal{H}_{d,\gamma}$ as

$$\mathcal{H}_{d,\gamma} = \mathcal{H}_{1,\gamma_1} \otimes \mathcal{H}_{1,\gamma_2} \otimes \ldots \otimes \mathcal{H}_{1,\gamma_d}\ ,$$

i.e. $\mathcal{H}_{d,\gamma}$ is the d-dimensional tensor product of the one-dimensional spaces $\mathcal{H}_{1,\gamma_j}$ with varying weights $\gamma_j > 0$, $j \in \{1,\ldots,d\}$ and weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$. The space $\mathcal{H}_{d,\gamma}$ is again a RKHS with reproducing kernel

$$K_{d,\gamma}(x,y) = \prod_{j=1}^d (1+\gamma_j\cdot\eta(x_j,y_j))\,.$$

If we define for a subset $u \subseteq D := \{1,\ldots,d\}$ that $\gamma_u := \prod_{j\in u}\gamma_j$ and $\gamma_\emptyset := 1$, then $K_{d,\gamma}(x,y)$ can be rewritten as

$$K_{d,\gamma}(x,y) = \sum_{u\subseteq\{1,\ldots,d\}} \gamma_u \prod_{j\in u}\eta(x_j,y_j)\,.$$

Remark: Weights of the form $\gamma_u := \prod_{j\in u} \gamma_j$ are called product weights. There are also other types of weights like order-dependent weights and finite-order weights. In this thesis, however, we will solely focus on product weights.

The inner product associated to $\mathcal{H}_{d,\gamma}$ takes the form

$$\langle f, g\rangle_{d,\gamma} = \sum_{u\subseteq\{1,\dots,d\}} \gamma_u^{-1} \int_{[0,1]^{|u|}} \frac{\partial^{|u|}}{\partial x_u} f(x_u; c) \cdot \frac{\partial^{|u|}}{\partial x_u} g(x_u; c)\, dx_u\,,$$

where x_u denotes the components x_j of x for which $j \in u$ and further $f(x_u; c)$ denotes the evaluation of f at the point $(x_u; c)$ defined by

$$(x_u; c)_j := \begin{cases} x_j & \text{, if } j \in u\,, \\ c_j & \text{, if } j \notin u\,, \end{cases} \qquad j \in \{1, \dots, d\}\,.$$

Here $c = (c_1, \dots, c_d)$ is the vector consisting of the anchor values of the spaces $\mathcal{H}_{1,\gamma_j}$.

The resulting space $\mathcal{H}_{d,\gamma}$ is called the weighted anchored Sobolev space of dimension d with associated weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$ and anchor vector $c = (c_1, \dots, c_d)$. It is a first example of a weighted reproducing kernel Hilbert space.

3.3 The weighted unanchored Sobolev space

In a similar fashion as before we can construct the unanchored weighted Sobolev space. Again, the one-dimensional building block is given by $\mathcal{H}_{1,\gamma}$, but now the corresponding inner product takes the form

$$\langle f, g\rangle_{1,\gamma} := \left(\int_0^1 f(x)\, dx\right) \cdot \left(\int_0^1 g(x)\, dx\right) + \frac{1}{\gamma} \int_0^1 f'(x) \cdot g'(x)\, dx\,.$$

The reproducing kernel of $\mathcal{H}_{1,\gamma}$ is again given by

$$K_{1,\gamma}(x, y) = 1 + \gamma \cdot \eta(x, y)\,,$$

but this time $\eta(x, y) = \frac{1}{2} B_2(|x - y|) + (x - \frac{1}{2})(y - \frac{1}{2})$ and $B_2(x) = x^2 - x + \frac{1}{6}$ is the Bernoulli polynomial of degree two.

To verify that $K_{1,\gamma}$ is indeed a reproducing kernel, we quickly check the reproducing property:

$$\begin{aligned} \langle f, K_{1,\gamma}(\cdot, y)\rangle_{1,\gamma} &= \left(\int_0^1 f(x)\, dx\right) \cdot \left(\int_0^1 K_{1,\gamma}(x, y)\, dx\right) + \frac{1}{\gamma} \int_0^1 f'(x) \cdot \frac{\partial}{\partial x} K_{1,\gamma}(x, y)\, dx \\ &= \left(\int_0^1 f(x)\, dx\right) \cdot \left(\int_0^1 1 + \gamma \cdot \eta(x, y)\, dx\right) + \frac{1}{\gamma} \int_0^1 f'(x) \cdot \gamma \cdot \frac{\partial}{\partial x} \eta(x, y)\, dx \\ &= \left(\int_0^1 f(x)\, dx\right) \cdot \left(1 + \gamma \cdot \int_0^1 \eta(x, y)\, dx\right) + \int_0^1 f'(x) \cdot \frac{\partial}{\partial x} \eta(x, y)\, dx \end{aligned}$$

Since $\eta(x, y) = \frac{1}{2} B_2(|x - y|) + (x - \frac{1}{2})(y - \frac{1}{2})$ we obtain further

$$\frac{\partial}{\partial x} \eta(x, y) = \frac{1}{2} \frac{\partial}{\partial x} B_2(|x - y|) + \left(y - \frac{1}{2}\right)\,.$$

Inserting the definition of the Bernoulli polynomial we have

$$B_2(|x-y|) = |x-y|^2 - |x-y| + \frac{1}{6} = x^2 - 2xy + y^2 - |x-y| + \frac{1}{6}$$

and thus

$$\frac{\partial}{\partial x} B_2(|x-y|) = 2x - 2y - \frac{\partial |x-y|}{\partial x},$$

where

$$\frac{\partial |x-y|}{\partial x} = \begin{cases} 1 & \text{, for } x \geq y\,, \\ -1 & \text{, for } x < y\,. \end{cases}$$

Combining the different expressions, we get as derivative of η

$$\frac{\partial}{\partial x}\,\eta(x,y) = \frac{1}{2}\cdot\left(2x - 2y - \frac{\partial |x-y|}{\partial x}\right) + \left(y - \frac{1}{2}\right) = x - \frac{1}{2} - \frac{1}{2}\cdot\frac{\partial |x-y|}{\partial x}\,.$$

Next, we calculate the integral of $\eta(x,y)$ over the interval $[0,1]$:

$$\begin{aligned} \int_0^1 \eta(x,y)\,dx &= \int_0^1 \frac{1}{2}B_2(|x-y|) + \left(x - \frac{1}{2}\right)\cdot\left(y - \frac{1}{2}\right)\,dx \\ &= \frac{1}{2}\int_0^1 B_2(|x-y|)\,dx + \left(y - \frac{1}{2}\right)\cdot \underbrace{\int_0^1 \left(x - \frac{1}{2}\right)\,dx}_{=0} = \frac{1}{2}\int_0^1 B_2(|x-y|)\,dx\,. \end{aligned}$$

Thus, we are left to calculate the integral of $B_2(|x-y|)$ on $[0,1]$:

$$\begin{aligned} \int_0^1 B_2(|x-y|)\,dx &= \int_0^1 x^2 - 2xy + y^2\,dx - \int_0^1 |x-y|\,dx + \frac{1}{6} \\ &= \left[\frac{1}{3}x^3 - x^2y + xy^2\right]_0^1 - \int_0^1 |x-y|\,dx + \frac{1}{6} = \frac{1}{3} - y + y^2 + \frac{1}{6} - \int_0^1 |x-y|\,dx \\ &= y^2 - y + \frac{1}{2} - \int_y^1 x - y\,dx - \int_0^y y - x\,dx \\ &= y^2 - y + \frac{1}{2} - \left[\frac{1}{2}x^2 - xy\right]_y^1 - \left[xy - \frac{1}{2}x^2\right]_0^y \\ &= y^2 - y + \frac{1}{2} - \frac{1}{2} + y + \frac{1}{2}y^2 - y^2 - y^2 + \frac{1}{2}y^2 = 0\,. \end{aligned}$$

Now, we insert the obtained expressions in the inner product equation and get

$$\begin{aligned}
\langle f, K_{1,\gamma}(\cdot, y)\rangle_{1,\gamma} &= \left(\int_0^1 f(x)\,dx\right)\cdot\left(1+\gamma\cdot\int_0^1 \eta(x,y)\,dx\right)+\int_0^1 f'(x)\cdot\frac{\partial}{\partial x}\,\eta(x,y)\,dx \\
&= \int_0^1 f(x)\,dx+\int_0^1 f'(x)\cdot\left(x-\frac{1}{2}-\frac{1}{2}\cdot\frac{\partial|x-y|}{\partial x}\right)\,dx \\
&= \int_0^1 f(x)+xf'(x)\,dx+\frac{1}{2}\left(-\int_0^1 f'(x)\,dx+\int_y^1 -f'(x)\,dx+\int_0^y f'(x)\,dx\right) \\
&= \int_0^1 f(x)\,dx+[xf(x)]_0^1-\int_0^1 f(x)\,dx+\frac{1}{2}\left(-f(1)-f(1)+f(y)+f(y)\right) \\
&= \frac{1}{2}\cdot(2f(y)-2f(1))+f(1)=f(y)\,.
\end{aligned}$$

Hence, $K_{1,\gamma}$ is indeed a reproducing kernel for the weighted unanchored Sobolev space.

The d-dimensional unanchored weighted Sobolev space is then given by the tensor product of the one-dimensional spaces $H_{1,\gamma}$, i.e.

$$\mathcal{H}_{d,\gamma}=\mathcal{H}_{1,\gamma_1}\otimes\mathcal{H}_{1,\gamma_2}\otimes\ldots\otimes\mathcal{H}_{1,\gamma_d}\ .$$

As before the reproducing kernel of $\mathcal{H}_{d,\gamma}$ is

$$K_{d,\gamma}(x,y)=\sum_{u\subseteq\{1,\ldots,d\}}\gamma_u\prod_{j\in u}\eta(x_j,y_j)\,,$$

with η as defined above. The corresponding inner product is now given by

$$\langle f,g\rangle_{d,\gamma}=\sum_{u\subseteq\{1,\ldots,d\}}\gamma_u^{-1}\int_{[0,1]^{|u|}}\left(\int_{[0,1]^{d-|u|}}\frac{\partial^{|u|}}{\partial x_u}f(x)\,dx_{D\setminus u}\right)\cdot\left(\int_{[0,1]^{d-|u|}}\frac{\partial^{|u|}}{\partial x_u}g(x)\,dx_{D\setminus u}\right)\,dx_u\,.$$

The weighted anchored and unanchored Sobolev spaces are two first examples of weighted RKHS, they are widely used in the analysis of the worst-case errors of QMC rules. Later we will use the reproducing kernels derived above and the worst-case error expression from chapter 2 to analyse the behaviour of the integration error in these spaces.

3.4 The weighted Korobov space

This function space is one of the most often used weighted reproducing kernel Hilbert spaces and will be essential for our error analysis in the subsequent chapters.

We consider one-periodic L^1-functions $f:[0,1]\to\mathbb{C}$ with absolutely convergent Fourier series, i.e. f has the Fourier series representation

$$f(x)=\sum_{h\in\mathbb{Z}}\hat{f}(h)\cdot e^{2\pi ihx}$$

with Fourier coefficients given by

$$\hat{f}(h) = \int_0^1 f(x) \cdot e^{-2\pi i h x} \, dx \, .$$

The space of all such functions equipped with the inner-product

$$\langle f, g \rangle_{1,\gamma} := \hat{f}(0) \cdot \overline{\hat{g}(0)} + \gamma^{-1} \sum_{h=-\infty}^{\infty}{}' \, |h|^{\alpha} \cdot \hat{f}(h) \cdot \overline{\hat{g}(h)}$$

forms the one-dimensional Hilbert space $H_{1,\gamma}$ with associated weight $\gamma > 0$. The parameter $\alpha > 1$ is referred to as smoothing parameter and the prime on the sum means that the term for $h = 0$ is omitted.

We can also rewrite the inner-product as

$$\langle f, g \rangle_{1,\gamma} = \sum_{h=-\infty}^{\infty} r(\gamma, h) \cdot \hat{f}(h) \cdot \overline{\hat{g}(h)} \, ,$$

where

$$r(\gamma, h) = \begin{cases} 1 & , \text{ if } h = 0 \, , \\ \gamma^{-1} \cdot |h|^{\alpha} & , \text{ if } h \neq 0 \, . \end{cases}$$

The obtained space $H_{1,\gamma}$ is in fact a RKHS with reproducing kernel

$$K_{1,\gamma}(x, y) = \sum_{h=-\infty}^{\infty} \frac{e^{2\pi i h (x-y)}}{r(\gamma, h)} \, .$$

To verify the reproducing property, we note that $K_{1,\gamma}(\cdot, y)$ is already given as Fourier series since

$$K_{1,\gamma}(x, y) = \sum_{h=-\infty}^{\infty} \frac{e^{2\pi i h (x-y)}}{r(\gamma, h)} = \sum_{h \in \mathbb{Z}} \frac{e^{-2\pi i h y}}{r(\gamma, h)} \cdot e^{2\pi i h x} \, .$$

The Fourier coefficients are therefore given by $\hat{K}_{1,\gamma}(\cdot, y)(h) = \frac{e^{-2\pi i h y}}{r(\gamma, h)}$ and we obtain further

$$\langle f, K_{1,\gamma}(\cdot, y) \rangle_{1,\gamma} = \sum_{h \in \mathbb{Z}} r(\gamma, h) \cdot \hat{f}(h) \cdot \frac{e^{2\pi i h y}}{r(\gamma, h)} = \sum_{h \in \mathbb{Z}} \hat{f}(h) \cdot e^{2\pi i h y} = f(y) \, .$$

Hence $K_{1,\gamma}$ is indeed the unique reproducing kernel of the space $H_{1,\gamma}$.

As in the previous section, we are particularly interested in the d-dimensional version of the weighted Korobov space. Again, we use a weight sequence $\gamma = \{\gamma_j\}_{j=1}^{d}$ of positive weights to define the space $H_{d,\gamma}$ as

$$H_{d,\gamma} = H_{1,\gamma_1} \otimes H_{1,\gamma_2} \otimes \ldots \otimes H_{1,\gamma_d} \, .$$

This function space is again a RKHS and contains all one-periodic L^1-functions defined on the d-dimensional unit cube $[0,1]^d$ which have an absolutely convergent Fourier series. The corresponding weighted inner-product is given by

$$\langle f, g\rangle_{d,\gamma} := \sum_{h\in\mathbb{Z}^d} \prod_{j=1}^{d} r(\gamma_j, h_j)\cdot \hat{f}(h)\cdot \overline{\hat{g}(h)}\,,$$

where the Fourier coefficients can be obtained via

$$\hat{f}(h) = \int_{[0,1]^d} f(x)\cdot e^{-2\pi i\langle h,x\rangle}\,dx \quad \text{with} \quad \langle h,x\rangle = \sum_{j=1}^{d} h_j\cdot x_j\,.$$

The reproducing kernel of $H_{d,\gamma}$ is given in a similar fashion as before, namely as the product of the one-dimensional kernels K_{1,γ_j}, where $j\in\{1,\dots,d\}$:

$$K_{d,\gamma}(x,y) = \prod_{j=1}^{d} K_{1,\gamma_j}(x_j,y_j) = \prod_{j=1}^{d}\sum_{h=-\infty}^{\infty} \frac{e^{2\pi i h(x_j-y_j)}}{r(\gamma_j,h)} = \prod_{j=1}^{d}\left(1+\gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j-y_j)}}{|h|^\alpha}\right).$$

The infinite sum in the kernel $K_{d,\gamma}$ is a bit unwieldy and hard to compute by hand. For certain values of the smoothing parameter α, however, the kernel takes a form that is easier to compute than the original form.

Remark:
We consider the Bernoulli polynomials B_k of degree k with $k\in\mathbb{N}$. Starting from $B_0 = 1$, the polynomials $B_k(x)$, $k>0$, are inductively defined on $[0,1]$ by:

(1) $\frac{d}{dx}B_k(x) = B_k'(x) = k\cdot B_{k-1}(x)$

(2) $\int_0^1 B_k(x) = 0\,.$

Using this definition, we can derive the Fourier coefficients of B_k and express B_k as Fourier series.

Lemma 3.1
For $\alpha>0$ an even integer the Bernoulli polynomial of degree α has the Fourier series representation

$$B_\alpha(x) = \frac{(-1)^{\frac{\alpha}{2}+1}\cdot\alpha!}{(2\pi)^\alpha}\cdot \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h x}}{|h|^\alpha}\,.$$

Proof. Let $\alpha>0$ be an even integer. We determine the Fourier coefficients $\hat{B}_\alpha(h)$ for $h\in\mathbb{Z}$:

(i) $\underline{h=0:}$

$$\int_0^1 B_\alpha(x)\cdot e^{-2\pi i h x}\,dx = \int_0^1 B_\alpha(x)\,dx \overset{(2)}{=} 0 \quad\Rightarrow\quad \hat{B}_\alpha(0) = 0$$

(ii) $\underline{h \neq 0:}$ Using integration by parts and the fundamental theorem of calculus, we obtain

$$\begin{aligned}
\int_0^1 B_\alpha(x) \cdot e^{-2\pi i h x}\, dx &\overset{IBP}{=} \left[B_\alpha(x) \cdot \frac{e^{-2\pi i h x}}{-2\pi i h} \right]_0^1 - \int_0^1 B'_\alpha(x) \cdot \frac{e^{-2\pi i h x}}{-2\pi i h}\, dx \\
&= \frac{1}{-2\pi i h} \cdot [B_\alpha(x)]_0^1 + \int_0^1 B'_\alpha(x) \cdot \frac{e^{-2\pi i h x}}{2\pi i h}\, dx \\
&= \frac{1}{-2\pi i h} \cdot \underbrace{\int_0^1 B'_\alpha(x)\, dx}_{= \alpha \cdot \int_0^1 B_{\alpha-1}(x)\, dx = 0} + \frac{1}{2\pi i h} \int_0^1 B'_\alpha(x) \cdot e^{-2\pi i h x}\, dx \\
&= \frac{1}{2\pi i h} \int_0^1 \alpha \cdot B_{\alpha-1}(x) \cdot e^{-2\pi i h x}\, dx \\
&= \frac{\alpha}{2\pi i h} \int_0^1 B_{\alpha-1}(x) \cdot e^{-2\pi i h x}\, dx
\end{aligned}$$

Note that $\int_0^1 B_{\alpha-1}(x)\, dx = 0$ for $\alpha \geq 2$ but not for $\alpha = 1$. Using the identity above, we obtain inductively (until $\alpha = 2$):

$$\int_0^1 B_\alpha(x) \cdot e^{-2\pi i h x}\, dx = \frac{\alpha}{2\pi i h} \cdot \frac{(\alpha-1)}{2\pi i h} \cdot \ldots \cdot \frac{2}{2\pi i h} \cdot \int_0^1 B_1(x) \cdot e^{-2\pi i h x}\, dx\,.$$

Further, we calculate the remaining integral and get

$$\begin{aligned}
\int_0^1 B_1(x) \cdot e^{-2\pi i h x}\, dx &= \int_0^1 \left(x - \frac{1}{2} \right) \cdot e^{-2\pi i h x}\, dx = \int_0^1 x \cdot e^{-2\pi i h x}\, dx \\
&= \left[x \cdot \frac{e^{-2\pi i h x}}{-2\pi i h} \right]_0^1 - \int_0^1 \frac{e^{-2\pi i h x}}{-2\pi i h}\, dx \\
&= -\frac{1}{2\pi i h} + \frac{1}{2\pi i h} \int_0^1 e^{-2\pi i h x}\, dx = -\frac{1}{2\pi i h}\,.
\end{aligned}$$

Hence, we obtain as Fourier coefficients for the Bernoulli polynomial B_α

$$\hat{B}_\alpha(h) = \int_0^1 B_\alpha(x) \cdot e^{-2\pi i h x}\, dx = \frac{\alpha}{2\pi i h} \cdot \frac{(\alpha-1)}{2\pi i h} \cdot \ldots \cdot \frac{2}{2\pi i h} \cdot \frac{-1}{2\pi i h} = \frac{-\alpha!}{(2\pi i h)^\alpha}\,.$$

Since $\alpha > 0$ is an even integer, the expression simplifies to

$$\hat{B}_\alpha(h) = \frac{\alpha!}{(2\pi)^\alpha} \cdot \frac{1}{h^\alpha} \cdot (-1) \cdot (-1)^{\frac{\alpha}{2}} = \frac{\alpha!}{(2\pi)^\alpha} \cdot \frac{1}{|h|^\alpha} \cdot (-1)^{\frac{\alpha}{2}+1}\,.$$

Using the obtained coefficients, we can now express $B_\alpha(x)$ as Fourier series by

$$B_\alpha(x) = \sum_{h \in \mathbb{Z}} \hat{B}_\alpha(h) \cdot e^{2\pi i h x} = \frac{(-1)^{\frac{\alpha}{2}+1} \cdot \alpha!}{(2\pi)^\alpha} \cdot {\sum_{h=-\infty}^{\infty}}' \frac{e^{2\pi i h x}}{|h|^\alpha}\,.$$

■

Using the previous Lemma, we can rewrite the reproducing kernel $K_{d,\gamma}$ as follows

$$K_{d,\gamma}(x,y) = \prod_{j=1}^{d} \left(1 + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha}\right) = \prod_{j=1}^{d} \left(1 + \gamma_j \cdot \frac{(2\pi)^\alpha \cdot (-1)^{\frac{\alpha}{2}+1}}{\alpha!} \cdot B_\alpha(|x_j - y_j|)\right).$$

The weighted Korobov space is another example for a weighted reproducing kernel Hilbert space. Later we will use the derived kernels $K_{d,\gamma}$ for both the weighted Sobolev and Korobov space to compute the worst-case error $e_{n,d}(Q_{n,d}, H)$ according to Theorem 2.6.

4 Tractability in weighted reproducing kernel Hilbert spaces

4.1 The concept of tractability

In this chapter we briefly discuss the notion of tractability and sum up recent results for the two reproducing kernel Hilbert spaces which were introduced in the previous chapter. The concept of tractability relates to the difficulty of a certain problem with respect to the dimension of the problem, in particular it addresses the question how fast the difficulty of a problem increases as the dimension grows.

As specified before, our problem is to approximate the integral given by

$$I(f) = \int_{[0,1]^d} f(x)\,dx$$

of a function f belonging to some normed function space H. Here, we restrict ourselves to approximations via families of Quasi-Monte Carlo rules of the form

$$Q_{n,d}(f) = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)\ . \tag{1}$$

As before, the worst-case error $e_{n,d}(Q_{n,d}, H)$ in the specified function space H is given by

$$e_{n,d}(Q_{n,d}, H) = \sup_{\|f\|_H \leq 1} |I(f) - Q_{n,d}(f)|\ .$$

We now define the tractability for QMC rules of the form (1) as follows.

Definition 4.1 (Minimal number)
For a given normed space H of functions $f : [0,1]^d \to \mathbb{R}$ and QMC rules of the form (1) we define the nth minimal number for the integration approximation problem as

$$n(\varepsilon, d) := \min\{n \in \mathbb{N} \,|\, \inf_{Q_{n,d}} e(Q_{n,d}, H) \leq \varepsilon\}\,,$$

where $d \in \mathbb{N}$ and $\varepsilon \in (0,1]$. The infimum is taken over the class of QMC rules $Q_{n,d}$ that use no more than n function values of f.

Definition 4.2 (Tractability)
The integration problem is said to be polynomially tractable if there exist constants $C > 0, p > 0$ and $q \geq 0$ such that

$$n(\varepsilon, d) \leq C \cdot d^q \cdot \varepsilon^{-p} \quad \text{for all } d \in \mathbb{N} \text{ and } \varepsilon \in (0,1]\,.$$

The integration problem is called strongly polynomially tractable if there are constants $C > 0, p > 0$ such that

$$n(\varepsilon, d) \leq C \cdot \varepsilon^{-p} \quad \text{for all } d \in \mathbb{N} \text{ and } \varepsilon \in (0,1]\,.$$

The infima of the constants p and q are called the ε-exponent and the d-exponent of tractability, respectively.

Both notions give us useful information about how many quadrature points have to be used to push the worst-case error below a specified error bound ε.

4.2 Tractability criteria

The notion of tractability plays an important role in weighted RKHS with associated weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$. Here, one can derive explicit criteria to decide whether a family of QMC rules is tractable or not. In the following section, we will briefly summarise important tractability results for the two weighted reproducing kernel Hilbert spaces we discussed earlier.

In [10], Sloan and Woźniakowski derived necessary and sufficient conditions for polynomial and strong polynomial tractability in weighted anchored and unanchored Sobolev spaces. Furthermore, they showed in [11] that similar equivalent conditions for (strong) polynomial tractability hold in the weighted Korobov function class.

Theorem 4.3
Consider the weighted anchored or unanchored Sobolev space with corresponding weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$. In this function space the associated integration problem with QMC rules of the form (1) is:

(i) strongly polynomially tractable if and only if $\sum_{j=1}^{\infty} \gamma_j < \infty$,

(ii) polynomially tractable if and only if $\limsup_{d\to\infty} \frac{\sum_{j=1}^d \gamma_j}{\log(d)} < \infty$.

Similarly, we have the following result for the weighted Korobov space.

Theorem 4.4
For the weighted Korobov space with weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$ the integration problem with QMC rules of the form (1) is:

(i) strongly polynomially tractable if and only if $\sum_{j=1}^{\infty} \gamma_j < \infty$,

(ii) polynomially tractable if and only if $\limsup_{d\to\infty} \frac{\sum_{j=1}^d \gamma_j}{\log(d)} < \infty$.

As before, we assumed that the weights are of product-weight type and that the weight sequence γ is positive and monotonically decreasing, i.e. $\gamma_1 \geq \gamma_2 \geq \ldots > 0$. We note that we have identical criteria for the two different types of tractability in both spaces. This indicates the close connection between the weighted Sobolev and Korobov space.

In particular, we obtain that in the weighted Sobolev and Korobov space, there exist quadrature rules $Q_{n,d}$ such that the worst-case error $e(Q_{n,d}, H_{d,\gamma})$ is bounded independently of the dimension d if and only if $\sum_{j=1}^{\infty} \gamma_j < \infty$. The notion of tractability will not further appear in the following work. Nevertheless, it is an important idea in the QMC theory which will affect our choice of weight sequences $\gamma = \{\gamma_j\}_{j=1}^d$ in the subsequent chapters.

5 The component-by-component construction

5.1 Lattice rules

The Quasi-Monte Carlo rules which we consider in this thesis are all of the form

$$Q_{n,d}(f) = \frac{1}{n}\sum_{i=0}^{n-1} f(x_i)\,,$$

where the quadrature points $x_0, \dots, x_{n-1} \in [0,1]^d$ are chosen deterministically. For the point set $P_n = \{x_0, \dots, x_{n-1}\}$ corresponding to the QMC rule $Q_{n,d}$, the integration error of $Q_{n,d}$ is closely connected with the discrepancy of P_n. The well-known Koksma-Hlawka inequality states that

$$\left| \int_{[0,1]^d} f(x)\,dx - Q_{n,d}(f) \right| \leq D_n^*(P_n) \cdot V(f)\,,$$

where $D_n^*(P_n)$ denotes again the star discrepancy of P_n given as

$$D_n^*(P_n) := \sup_{y \in [0,1]^d} \left| \frac{1}{n} A([0,y], P_n) - \lambda_d([0,y]) \right|$$

and $V(f)$ is the variation of the function f in the sense of Hardy and Krause.

For our weighted reproducing kernel Hilbert spaces there exists a similar version of the Koksma-Hlawka inequality. In [10], Sloan and Woźniakowski defined the notion of weighted discrepancy and proved a "weighted" Koksma-Hlawka inequality (for more details see [10, p. 9,10]). The weighted Koksma-Hlawka inequality states that for a quadrature rule $Q_{n,d}$ with quadrature points $x_0, \dots, x_{n-1}$ and a function f of some weighted reproducing kernel Hilbert space H we have

$$\left| \int_{[0,1]^d} f(x)\,dx - Q_{n,d}(f) \right| \leq \mathrm{disc}_\gamma(\{x_i\}) \cdot \|f\|_{d,\gamma}\,,$$

where the weighted discrepancy $\mathrm{disc}_\gamma(\{x_i\})$ of the quadrature points x_i is defined as

$$\mathrm{disc}_\gamma(\{x_i\}) := \left(\sum_{\emptyset \neq u \subseteq D} \gamma_u \int_{[0,1]^{|u|}} \Delta^2(x_u, 1)\,dx_u \right)^{1/2}$$

and

$$\|f\|_{d,\gamma} = \left(\sum_{\emptyset \neq u \subseteq D} \gamma_u \int_{[0,1]^{|u|}} \left| \frac{\partial^{|u|}}{\partial x_u} f(x_u, 1) \right|^2 dx_u \right)^{1/2}$$

is the norm of f in the weighted anchored Sobolev space with weight sequence γ and anchor vector c.

The above inequality indicates that the choice of quadrature points is crucial to obtain a small integration error. Ideally, we want to find a point set P_n or a sequence of quadrature points x_i such that the weighted discrepancy $\mathrm{disc}_\gamma(\{x_i\})$ is minimized. There are many ways to determine the points x_i for a Quasi-Monte Carlo rule, in this thesis, however, we will focus on lattice rules, in particular rank-one lattice rules.

Definition 5.1 (Rank-one lattice rule)
A rank-one lattice rule is a Quasi-Monte Carlo rule with underlying point set $P_n \subseteq [0,1]^d$ of the form

$$P_n = \left\{ \left\{ \frac{k \cdot z}{n} \right\} \,\middle|\, 0 \leq k < n \right\} \subseteq [0,1]^d ,$$

where $k \cdot z$ denotes the scalar multiplication of $k \in \mathbb{Z}$ with the integer vector z and $\{\cdot\}$ denotes the component-wise fractional part of the term inside the braces.

Further, we call $z \in \mathbb{Z}^d$ the generating vector of the rank-one lattice rule.

In the following section we are interested in the worst-case error of rank-one lattice rules. Since any rank-one lattice rule is uniquely defined by its generating vector z, we denote the corresponding worst-case error of a QMC rule $Q_{n,d}$ with point set P_n generated by z by $e_{n,d}(z, H)$ or simply $e_{n,d}(z)$. Moreover, the components of z are restricted to the set

$$Z_n := \{0 < \hat{z} < n \mid \gcd(\hat{z}, n) = 1\} ,$$

i.e. we have $z \in Z_n^d$.

Firstly, we consider functions from the weighted Korobov space. Recall that the reproducing kernel in the weighted Korobov space is given by

$$K_{d,\gamma}(x,y) = \prod_{j=1}^{d} \left(1 + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha} \right) ,$$

or in a more general form

$$K_{d,\gamma}(x,y) = \prod_{j=1}^{d} \left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha} \right) .$$

Using the latter kernel and the worst-case error expression from Theorem 2.6, we obtain the following Lemma.

Lemma 5.2
The squared worst-case error for a rank-one lattice rule with generating vector z in the weighted Korobov space is given as

$$e_{n,d}^2(z) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h k z_j / n}}{|h|^\alpha} \right) .$$

Proof. By Theorem 2.6 and with the kernel $K_{d,\gamma}$ we obtain the following expressions:

(a)

$$\int_{[0,1]^d} \int_{[0,1]^d} K_{d,\gamma}(x,y)\, dx\, dy = \int_{[0,1]^{d-1}} \int_{[0,1]^{d-1}} \prod_{\substack{k=1 \\ k \neq j}}^{d} \left(\beta_k + \gamma_k \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_k - y_k)}}{|h|^\alpha} \right) \cdot \underbrace{\left(\int_0^1 \int_0^1 \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha} \, dx_j \, dy_j \right)}_{=: I_1} dx_{D\setminus\{j\}}\, dy_{D\setminus\{j\}}$$

Now, we consider the separated, one-dimensional integrals in the second line above

$$\begin{aligned} I_1 &= \int_0^1 \int_0^1 \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha} \, dx_j \, dy_j \\ &= \int_0^1 \beta_j + \gamma_j \int_0^1 \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{|h|^\alpha} \, dx_j \, dy_j \\ &= \beta_j + \gamma_j \left(\int_0^1 \sum_{h=-\infty}^{\infty}{}' \frac{1}{|h|^\alpha} \int_0^1 e^{2\pi i h(x_j - y_j)} \, dx_j \, dy_j \right) \\ &= \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{1}{|h|^\alpha} \int_0^1 e^{-2\pi i h y_j} \underbrace{\int_0^1 e^{2\pi i h x_j} \, dx_j}_{=0 \ \forall \, h} \, dy_j = \beta_j \, . \end{aligned}$$

Inductively, we obtain for the original term above that

$$\int_{[0,1]^d} \int_{[0,1]^d} K_{d,\gamma}(x,y) \, dx \, dy = \prod_{j=1}^{d} \beta_j \, .$$

(b) Here, we consider at first the term

$$\int_{[0,1]^d} K_{d,\gamma}(x, x_k) = \int_{[0,1]^d} K_{d,\gamma} \left(x, \left\{ \frac{k \cdot z}{n} \right\} \right) dx \quad \text{for} \quad k = 0, 1, \ldots, n-1 \, .$$

As in (a) we can split up the d-dimensional integral into one-dimensional parts given as

$$\begin{aligned} I_2 &:= \int_0^1 \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h \left(x_j - \left\{ \frac{k \cdot z_j}{n} \right\} \right)}}{|h|^\alpha} \, dx_j = \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{1}{|h|^\alpha} \int_0^1 e^{2\pi i h \left(x_j - \left\{ \frac{k \cdot z_j}{n} \right\} \right)} \, dx_j \\ &= \beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{1}{|h|^\alpha} \cdot e^{-2\pi i h \left\{ \frac{k \cdot z_j}{n} \right\}} \underbrace{\int_0^1 e^{2\pi i h x_j} \, dx_j}_{=0 \ \forall \, h} = \beta_j \, . \end{aligned}$$

Inductively, we obtain for the original term above that

$$\int_{[0,1]^d} K_{d,\gamma}(x, x_k) = \prod_{j=1}^{d} \beta_j \quad \text{for} \quad k = 0, 1, \ldots, n-1 \, .$$

(c) At last, we calculate the term $K_{d,\gamma}(x_i, x_k)$ with

$$K_{d,\gamma}(x_i, x_k) = \prod_{j=1}^{d} \left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h \left(\left\{ \frac{i \cdot z_j}{n} \right\} - \left\{ \frac{k \cdot z_j}{n} \right\} \right)}}{|h|^\alpha} \right) = \prod_{j=1}^{d} \left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h (i-k) z_j / n}}{|h|^\alpha} \right) .$$

This yields that for the double sum $\sum_{i=0}^{n-1}\sum_{k=0}^{n-1} K_{d,\gamma}(x_i,x_k)$ the term $(i-k)$ runs through all values from $-n+1$ to $n-1$. Since $e^{2\pi i h a/n} = e^{2\pi i h b/n}$ if $a \equiv b \mod n$, where $a,b \in \mathbb{Z}$, we see that

$$\sum_{i=0}^{n-1}\sum_{k=0}^{n-1} K_{d,\gamma}(x_i,x_k) = n \cdot \sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h k z_j/n}}{|h|^\alpha}\right).$$

Via Theorem 2.6 we can calculate the squared worst-case error $e^2_{n,d}(z)$ as follows

$$\begin{aligned} e^2_{n,d}(z) &= \int_{[0,1]^d}\int_{[0,1]^d} K_{d,\gamma}(x,y)\,dx\,dy - \frac{2}{n}\sum_{k=0}^{n-1}\int_{[0,1]^d} K_{d,\gamma}(x_k,x)\,dx + \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{k=0}^{n-1} K_{d,\gamma}(x_i,x_k) \\ &= \prod_{j=1}^{d}\beta_j - \frac{2}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\beta_j + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h k z_j/n}}{|h|^\alpha}\right) \\ &= -\prod_{j=1}^{d}\beta_j + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h k z_j/n}}{|h|^\alpha}\right). \end{aligned}$$

■

Next, we change the considered function space from the weighted Korobov space to the weighted anchored Sobolev space from the previous chapters. Here, the kernel $K_{d,\gamma}$ takes the form

$$K_{d,\gamma}(x,y) = \prod_{j=1}^{d}(1+\gamma_j \cdot \eta(x,y)).$$

Associated with any reproducing kernel $K_d(x,y)$ is the so-called shift-invariant kernel defined by

$$K^*_d(x,y) := \int_{[0,1]^d} K_d(\{x+\Delta\},\{y+\Delta\})\,d\Delta.$$

The shift-invariant kernel $K^*_{d,\gamma}$ corresponding to $K_{d,\gamma}$ is given by (see [8] for details)

$$K^*_{d,\gamma}(x,y) = \prod_{j=1}^{d}\left[1+\gamma_j\cdot\left(B_2(|x_j-y_j|)+c_j^2-c_j+\frac{1}{3}\right)\right],$$

where $c=(c_1,\ldots,c_d)$ is the anchor vector of the Sobolev space. By Lemma 3.1 the Bernoulli polynomial of degree two can be written as

$$B_2(x) = \frac{1}{2\pi^2}\sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h x}}{h^2}.$$

Therefore, we can rewrite the shift-invariant kernel above as

$$K^*_{d,\gamma}(x,y) = \prod_{j=1}^{d}\left(\hat{\beta}_j + \hat{\gamma}_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j-y_j)}}{h^2}\right),$$

with $\hat{\beta}_j = 1+\gamma_j(c_j^2 - c_j + \frac{1}{3})$ and $\hat{\gamma}_j = \frac{\gamma_j}{2\pi^2}$. We see that this is in fact the reproducing kernel for the weighted Korobov space with smoothing parameter $\alpha = 2$, weight sequence $\gamma_j = \hat{\gamma}_j$ and constant terms $\hat{\beta}_j$. Similarly, the shift-invariant kernel $K^*_{d,\gamma}$ for the weighted unanchored Sobolev space is given by

$$K^*_{d,\gamma}(x,y) = \prod_{j=1}^{d} \left[\, 1 + \gamma_j \cdot B_2(|x_j - y_j|) \,\right] ,$$

which can be rewritten as

$$K^*_{d,\gamma}(x,y) = \prod_{j=1}^{d} \left(\hat{\beta}_j + \hat{\gamma}_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j - y_j)}}{h^2} \right) ,$$

with $\hat{\beta}_j = 1$ and $\hat{\gamma}_j = \frac{\gamma_j}{2\pi^2}$. Again, this kernel corresponds to the reproducing kernel of the weighted Korobov space for $\alpha = 2$ and associated sequences $\hat{\beta}, \hat{\gamma}$.

The shift-invariant kernels can be used to analyse the error behaviour of so-called randomly shifted rank-one lattice rules which are randomized QMC rules with underlying point set of the form

$$\left\{ \left\{ \frac{k \cdot z}{n} + \Delta_m \right\} \middle| \, k = 0, 1, \ldots, n-1, m = 1, \ldots, q \right\} ,$$

where the shifts Δ_m are independently uniformly distributed on $[0,1]^d$. The associated squared shift-averaged worst-case error for the weighted anchored Sobolev space is then given by

$$\hat{e}^2_{n,d}(z) = -\prod_{j=1}^{d} \left(1 + \gamma_j \left(c_j^2 - c_j + \frac{1}{3}\right)\right) + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \left[B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right) + c_j^2 - c_j + \frac{1}{3}\right]\right) .$$

Similarly, we obtain for the weighted unanchored Sobolev space that

$$\hat{e}^2_{n,d}(z) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right) .$$

More details for the construction of and search for randomly shifted rank-one lattice rules can be found in [2] and [9]. The above relations between the weighted Sobolev and Korobov space emphasize the connection between the two RKHS.

With the derived worst-case error expressions we have a direct connection between the generating vector z of a lattice rule and the integration error. Naturally, we aim to find good generating vectors such that the worst-case error $e_{n,d}(z)$ is as small as possible. Due to the close connection between the worst-case errors of the weighted Sobolev and Korobov space, we will mostly consider the error term of the weighted unanchored Sobolev space.

5.2 The component-by-component construction

Our goal is to find a generating vector $z \in Z_n^d$ such that the corresponding worst-case error $e_{n,d}(z)$ is as small as possible. We can formulate this goal as the following minimization problem:

$$\begin{aligned} &\text{minimize} && e^2_{n,d}(z) \\ &\text{subject to} && z \in Z_n^d = \{0 < \hat{z} < n | \, \gcd(\hat{z}, n) = 1\}^d. \end{aligned} \tag{MP}$$

For n a prime number Z_n simplifies to $Z_n = \{1, 2, \ldots, n-1\}$. For simplicity's sake, we will restrict ourselves to quadrature rules with a prime number n of quadrature points. The total number of feasible generating vectors $z \in Z_n^d$ is finite, since $|Z_n^d| = (n-1)^d$. However, an exhaustive search over all $(n-1)^d$ possible vectors is infeasible as the number of choices grows exponentially with d.

In order to construct good lattice rules, one has to use a different method that finds generating vectors z with a small worst-case error $e_{n,d}(z)$. An interesting technique to select suitable generating vectors is the so-called component-by-component construction, which has been intensively studied in the last years. In essence, the component-by-component algorithm (CBC algorithm) chooses the components z_i of the generating vector one component at a time, while keeping all previously chosen components fixed. In each step of the selection process the algorithm selects the value $z_j \in Z_n$ with $j \in \{1, \ldots, d\}$ which minimizes the corresponding error term $e_{n,j}^2(z_1, \ldots, z_j)$. The pseudocode of the component-by-component algorithm is given below.

Algorithm 1 Component-by-component algorithm for weighted tensor-product RKHS

for $d = 1$ **to** $d_{\mathbf{max}}$ **do**
  for all $z_d \in Z_n$ **do**
    $$e_{n,d}^2(z_1, z_2, \ldots, z_{d-1}, z_d) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right)$$
  end for
  $z_d = \underset{z \in Z_n}{\operatorname{argmin}}\ e_{n,d}^2(z_1, z_2, \ldots, z_{d-1}, z)$
end for

Here, ω denotes some real-valued function which is associated with the respective reproducing kernel of the considered RKHS. If, for example, the smoothing parameter α happens to be an even integer, the function ω becomes the Bernoulli polynomial of degree α scaled by a factor (compare to 3.1 and 5.2). Further, we will call every generating vector $z \in Z_n^d$ that has been constructed by Algorithm 1 a **CBC vector**.

Remark: In the following chapters we will use squared worst-case errors of the general form

$$e_{n,d}^2(z_1, \ldots, z_d) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right)$$

but also of the special form

$$e_{n,d}^2(z_1, \ldots, z_d) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) .$$

Furthermore, we mainly focus our analysis on the error expression given by

$$e_{n,d}^2(z_1, \ldots, z_d) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot B_2 \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) ,$$

which corresponds to the weighted unanchored Sobolev space with parameters $\beta_j = 1$ for all $j = 1, \ldots, d$.

In order to assess the computational cost of the component-by-component algorithm, we examine Algorithm 1 more closely. At first we rewrite the error term

$$e^2_{n,d}(z_1,\ldots,z_d) = -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(1+\gamma_j\cdot\omega\left(\left\{\frac{k\cdot z_j}{n}\right\}\right)\right)$$

that has to be calculated in each step of the algorithm as

$$e^2_{n,d}(z_1,\ldots,z_d) = -1 + \frac{1}{n}\sum_{k=0}^{n-1} p_{d-1}(k)\cdot\left(1+\gamma_d\cdot\omega\left(\left\{\frac{k\cdot z_d}{n}\right\}\right)\right),$$

where $p_{d-1}(k)$ is recursively defined as $p_{d-1}(k) = p_{d-2}(k)\cdot\left(1+\gamma_{d-1}\cdot\omega\left(\left\{\frac{k\cdot z_{d-1}}{n}\right\}\right)\right)$ and $p_0(k)=1$ for all $k\in\{0,1,\ldots,n-1\}$. In each step of the component-by-component algorithm, we have to calculate $e^2_{n,d}(z_1,z_2,\ldots,z_{d-1},z)$ for all $z\in Z_n$. The major part of this calculation can be carried out as one matrix-vector product for all $z\in Z_n$ via

$$v_d(z) := \sum_{k=0}^{n-1} p_{d-1}(k)\cdot\omega\left(\left\{\frac{k\cdot z}{n}\right\}\right) = (\Omega_n\cdot p_{d-1})(z),$$

with p_{d-1} the n-dimensional vector defined above and Ω_n a matrix defined as

$$\Omega_n := \left[\omega\left(\left\{\frac{k\cdot z}{n}\right\}\right)\right]_{\substack{z=1,\ldots,n-1\\ k=0,\ldots,n-1}}.$$

Then we have that $e^2_{n,d}(z_1,\ldots,z_{d-1},z) = -1 + \frac{1}{n}\sum_{k=0}^{n-1} p_{d-1}(k) + \frac{\gamma_d}{n}\cdot v_d(z)$. The complexity of the matrix-vector product is $\mathcal{O}(n^2)$ and so the time complexity to calculate the squared worst-case error $e^2_{n,d}(z_1,\ldots,z_{d-1},z)$ for all $z\in Z_n$ in each step of the algorithm is $\mathcal{O}(n^2)$. Thus, the complexity of the CBC algorithm in d dimensions is $\mathcal{O}(d\,n^2)$, where $\mathcal{O}(n)$ memory is needed to store the n-dimensional vector p_{d-1} which is updated in each step.

To stress the importance of the component-by-component construction, we sum up some results about the theory of rank-one lattice rules in weighted reproducing kernel Hilbert spaces. According to those theoretical results, there are lattice rules which achieve optimal convergence rates. Sloan and Woźniakowski showed in [11] that the optimal rate of convergence for multivariate integration in weighted Korobov spaces is $\mathcal{O}(n^{-\alpha/2+\delta})$, where α is the smoothing parameter and δ is some positive number. More precisely, they showed that if

$$\sum_{j=1}^{\infty}\gamma_j^{1/\alpha} < \infty,$$

then for n prime and $\delta>0$ there exist lattice rules such that the error is bounded independently of d like

$$e_{n,d}(z) \le C(\delta)\cdot n^{-\alpha/2+\delta}.$$

Similar results were obtained by Sloan and Woźniakowski for shifted rank-one lattice rules in the weighted Sobolev space (compare again to [11]). They showed that if the weights satisfy the

condition

$$\sum_{j=1}^{\infty} \gamma_j^{1/2} < \infty \,,$$

then for n prime and $\delta > 0$ there exist shifted lattice rules such that the corresponding worst-case error is bounded like

$$e_{n,d}(z) \leq C(\delta) \cdot n^{-1+\delta} \,.$$

The convergence rate $\mathcal{O}(n^{-1+\delta})$ of integration in the weighted Sobolev space is further the optimal rate of convergence. However, the arguments of Sloan and Woźniakowski were nonconstructive in that they proved the existence of lattice rules achieving the optimal rate of convergence but did not demonstrate how such lattice rules could be constructed.

Later, Kuo showed in [2] that the component-by-component construction introduced above actually generates lattice rules which achieve these optimal rates of convergence in weighted Korobov and Sobolev spaces, whose existence was before only proven theoretically. We summarise the obtained results for the component-by-component construction in weighted Korobov and Sobolev spaces.

Theorem 5.3 (Kuo [2])
Let n be a prime number and let z be constructed component-by-component as in Algorithm 1. Then this generating vector z satisfies

$$e_{n,d}(z) \leq C_d(\delta)\, n^{-\frac{\alpha}{2}+\delta} e_{0,d} \quad \text{for all} \quad 0 < \delta \leq \frac{\alpha - 1}{2}$$

where

$$C_d(\delta) = 2^{\frac{\alpha}{2}-\delta} \prod_{j=1}^{d} \left[1 + 2 \left(\frac{\gamma_j}{\beta_j} \right)^{\frac{1}{\alpha - 2\delta}} \zeta\left(\frac{\alpha}{\alpha - 2\delta} \right) \right]^{\frac{\alpha}{2}-\delta} \quad \text{and} \quad e_{0,d} = \prod_{j=1}^{d} \beta_j^{\frac{1}{2}} \,.$$

Moreover, if

$$\sum_{j=1}^{\infty} \left(\frac{\gamma_j}{\beta_j} \right)^{\frac{1}{\alpha - 2\delta}} < \infty$$

then

$$C_d(\delta) \leq C_\infty(\delta) < \infty \,,$$

that is, $e_{n,d}(z)$ is $\mathcal{O}(n^{-\alpha/2+\delta})$ for $\delta > 0$, with the implied constant independent of d. Hence, the ε-exponent of strong tractability is $2/\alpha$.

Here the corresponding squared worst-case error in the weighted Korobov space is given in the generalized form

$$e_{n,d}^2(z) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h k z_j / n}}{|h|^\alpha} \right) .$$

A similar result holds for shifted lattice rules in the weighted Sobolev space with associated squared shift-averaged worst-case error given as

$$\hat{e}_{n,d}^2(z) = -\prod_{j=1}^{d} \left(\beta_j + \gamma_j \left(c_j^2 - c_j + \frac{1}{3} \right) \right) + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \left[B_2\left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) + c_j^2 - c_j + \frac{1}{3} \right] \right) .$$

Theorem 5.4 (Kuo [2])
Let n be a prime number and let z be constructed component-by-component as in Algorithm 1. We have

$$\hat{e}_{n,d}(z) \leq C_d(\delta)\, n^{-1+\delta} e_{0,d} \quad \text{for all} \quad 0 < \delta \leq \frac{1}{2}$$

where

$$C_d(\delta) = 2^{1-\delta} \prod_{j=1}^{d} \left[1 + 2 \left(\frac{\frac{\gamma_j}{2\pi^2}}{\beta_j + \gamma_j \left(c_j^2 - c_j + \frac{1}{3} \right)} \right)^{\frac{1}{2(1-\delta)}} \zeta\left(\frac{1}{1-\delta} \right) \right]^{1-\delta},$$

and

$$e_{0,d} = \prod_{j=1}^{d} \left(\beta_j + \gamma_j \left(c_j^2 - c_j + \frac{1}{3} \right) \right)^{\frac{1}{2}}.$$

Moreover, if

$$\sum_{j=1}^{\infty} \left(\frac{\gamma_j}{\beta_j} \right)^{\frac{1}{2(1-\delta)}} < \infty$$

then

$$C_d(\delta) \leq C_\infty(\delta) < \infty$$

that is, $e_{n,d}(z)$ is $\mathcal{O}(n^{-1+\delta})$ for $\delta > 0$, with the implied constant independent of d. Hence, the ε-exponent of strong tractability is 1.

In 2006, Nuyens and Cools came up with a fast version of the component-by-component construction which reduced the cost of the algorithm to only $\mathcal{O}(d\, n \log(n))$ operations (compare to [5]). The key point in this fast construction was the reduction of the cost of the matrix-vector product calculation to only $\mathcal{O}(n \log(n))$ operations by using fast Fourier transformation and inverse fast Fourier transformation. Furthermore, they partitioned the matrix Ω_n into blocks of circulant or block circulant matrices using algebraic permutation theory. We provide a short pseudocode of the fast algorithm below (compare to [1, p. 82]). For more details we refer the interested reader to [5].

Algorithm 2 Fast CBC: matrix-vector form with permuted matrix

$p_0 = \mathbf{1}$
$e_0^2 = 0$
for $d = 1$ **to** $d_{\max}$ **do**
  $\mathbf{e_d^2}^{\langle g \rangle} = (\mathbf{1} + \gamma_d \boldsymbol{\beta})\, e_{d-1}^2 + \frac{\gamma_d}{n} \boldsymbol{\Omega_n}^{\langle g \rangle} \mathbf{p_{d-1}}$ ▷ **compute - use FFT**
  $z_d = \operatorname{argmin}_{z \in Z_n} \; e_{n,d}^2(z_1, \ldots, z_{d-1}, z)$ ▷ **select - pick the correct index**
  $e_d^2 = e_d^2(z_d)$ ▷ **set**
  $\mathbf{p_d} = \left(\mathbf{1} + \gamma_d (\boldsymbol{\Omega_n}^{\langle g \rangle}(z_d, :) + \boldsymbol{\beta}) \right) .* \mathbf{p_{d-1}}$ ▷ **update**
end for

Thus, we obtain that it is possible to construct rank-one lattice rules by the component-by-component algorithm in a fast manner.

5.3 The minimization problem

We saw that the component-by-component construction generates lattice rules such that the corresponding worst-case error exhibits the optimal rate of convergence in the respective function spaces. However, the CBC algorithm does not guarantee to solve the minimization problem (MP) with respect to the error value formulated at the beginning of the last section. Therefore, we want to examine how good the generating vectors z obtained by the CBC algorithm actually are in terms of the magnitude of the associated error $e_{n,d}(z)$.

Firstly, we perform some numerical experiments for the component-by-component algorithm. As setting we choose the weighted unanchored Sobolev space of dimension d with corresponding squared worst-case error for rank-one lattice rules given by

$$e^2_{n,d}(z) = -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(1+\gamma_j \cdot B_2\left(\left\{\frac{k\cdot z_j}{n}\right\}\right)\right),$$

where $B_2(x) = x^2 - x + \frac{1}{6}$ is the Bernoulli polynomial of degree two and $\gamma = \{\gamma_j\}_{j=1}^d$ is the associated weight sequence. Furthermore, we only consider lattice rules with a prime number n of quadrature points.

For a small dimension of $d = 4$ we compute the generating vectors z of a lattice rule with both the CBC algorithm as in Algorithm 1 and the fast CBC version by Nuyens and Cools (compare to Algorithm 2). Moreover, we compute the best generating vector amongst all $(n-1)^d$ possible generating vectors by an exhaustive search. The corresponding worst-case errors are then calculated and compared. In particular, we investigate the error behaviour for different weight sequences $\gamma = \{\gamma_j\}_{j=1}^d$ and different numbers n of quadrature points.

Remark: Note that $B_2(\cdot)$ is symmetric around $x = \frac{1}{2}$, i.e. $B_2(x) = B_2(1-x)$ for $x \in [0,1]$. Thus, we have for $z \in Z_n$:

$$\begin{aligned} B_2\left(\left\{\frac{k\cdot z}{n}\right\}\right) &= B_2\left(1-\left\{\frac{k\cdot z}{n}\right\}\right) = B_2\left(\left\{1-\frac{k\cdot z}{n}\right\}\right) \\ &= B_2\left(\left\{k-\frac{k\cdot z}{n}\right\}\right) = B_2\left(\left\{\frac{k\cdot(n-z)}{n}\right\}\right), \end{aligned}$$

where $k \in \{1,\dots,n-1\}$ is arbitrarily chosen. Hence, we obtain that for $\bar{z} = (n-z_1,\dots,n-z_d)$ and $z \in Z_n^d$

$$e^2_{n,d}(\bar{z}) = -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(1+\gamma_j \cdot B_2\left(\left\{\frac{k\cdot(n-z_j)}{n}\right\}\right)\right) = e^2_{n,d}(z)\,.$$

This also implies that for every generating vector $z = (z_1,\dots,z_d) \in Z_n^d$ there exist $2^d - 1$ different generating vectors $\bar{z} \in Z_n^d$ which have the same worst-case error $e_{n,d}(z)$ as z. These 2^d-1 different vectors are those whose components $\bar{z}_j$ are either z_j or $n - z_j$ (but not all $\bar{z}_j = z_j$). Thus, we only have to consider generating vectors z with components in Z_m with $m := \frac{(n-1)}{2}$. Therefore it is possible to restrict the search to the set Z_m^d of cardinality $\frac{(n-1)^d}{2^d} = m^d$ instead of $(n-1)^d$.

We have the following notations:

$$\bar{z} = \text{ fast CBC vector}, z^* = \text{ CBC vector}, z_{\text{ex}} = \text{ vector obtained by exhaustive search}$$

Further P_1 and P_2 indicate the ranking of $\bar{z}$ and z^*, respectively, based on their worst-case error amongst all $\left(\frac{n-1}{2}\right)^d$ possible generating vectors in Z_m^d, e.g. $P_2 = 2$ means that z^* is the second-best generating vector for a lattice rule with n points in the respective RKHS.

Numerical Experiments:

Table 1: $d = 4, \beta_j = 1, \gamma_j = 1/j^2$

n	$e_{n,d}(\bar{z})$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$	P_1	P_2
101	6.6448e-03	6.6448e-03	6.5650e-03	34	34
127	5.3239e-03	5.3093e-03	5.2944e-03	16	5
139	4.8861e-03	4.8921e-03	4.8718e-03	6	11
151	4.4991e-03	4.4795e-03	4.4795e-03	8	1
181	3.7322e-03	3.7425e-03	3.7322e-03	1	4
199	3.4485e-03	3.4485e-03	3.4274e-03	13	13

Table 2: $d = 4, \beta_j = 1, \gamma_j = 1/j^3$

n	$e_{n,d}(\bar{z})$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$	P_1	P_2
101	5.1275e-03	5.1093e-03	5.1093e-03	29	1
127	4.1203e-03	4.1161e-03	4.1145e-03	10	2
139	3.7623e-03	3.7623e-03	3.7606e-03	5	5
151	3.4531e-03	3.4475e-03	3.4475e-03	4	1
181	2.8784e-03	2.8784e-03	2.8750e-03	3	3
199	2.6358e-03	2.6310e-03	2.6310e-03	11	1

Table 3: $d = 4, \beta_j = 1, \gamma_j = (0.9)^j$

n	$e_{n,d}(\bar{z})$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$	P_1	P_2
101	1.6858e-02	1.7242e-02	1.6798e-02	7	336
127	1.3681e-02	1.3681e-02	1.3495e-02	82	82
139	1.2850e-02	1.2850e-02	1.2633e-02	204	204
151	1.1957e-02	1.1991e-02	1.1738e-02	154	205
181	1.0096e-02	1.0120e-02	9.7840e-03	139	166
199	9.2654e-03	9.2822e-03	9.0730e-03	161	187

Table 4: $d = 4, \beta_j = 1, \gamma_j = (0.5)^j$

n	$e_{n,d}(\bar{z})$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$	P_1	P_2
101	5.2692e-03	5.2692e-03	5.2183e-03	29	29
127	4.1826e-03	4.1992e-03	4.1760e-03	2	18
139	3.8641e-03	3.8641e-03	3.8578e-03	7	7
151	3.5605e-03	3.5484e-03	3.5484e-03	4	1
181	2.9626e-03	2.9586e-03	2.9501e-03	6	3
199	2.7169e-03	2.7169e-03	2.7106e-03	5	5

Table 5: $d = 4, \beta_j = 1, \gamma_j = (0.1)^j$

n	$e_{n,d}(\bar{z})$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$	P_1	P_2
101	1.3615e-03	1.3615e-03	1.3614e-03	4	4
127	1.0842e-03	1.0842e-03	1.0842e-03	1	1
139	9.9031e-04	9.9031e-04	9.9031e-04	2	1
151	9.1135e-04	9.1136e-04	9.1135e-04	1	2
181	7.6017e-04	7.6017e-04	7.6017e-04	1	1
199	6.9166e-04	6.9166e-04	6.9166e-04	1	3

Observations: Firstly, we pay attention to the decay of the used weight sequences $\gamma = \{\gamma_j\}_{j=1}^d$. Examining our results, we see that there is a direct connection between the decay of the weight sequence γ and the performance of the CBC algorithm in the respective function space. The faster the weights decline, the better the component-by-component construction works with regard to the minimization problem. If we consider for example the weights $\gamma_j = 1/j^3$ and $\gamma_j = (0.1)^j$, we observe that the generating vector z^*, constructed by the classic CBC algorithm, is for all values of n amongst the five best generating vectors. Moreover, we sometimes even obtain that $z^* = z_{\text{ex}}$, i.e. the CBC vector z^* solves the minimization problem (MP) stated above. If, on the other hand, the weight sequence is decaying slowly (as for $\gamma_j = (0.9)^j$), then there is a large number of generating vectors z which attain a smaller worst-case error $e_{n,d}(z)$ than z^*.

Further, we notice that the generating vector $\bar{z}$ obtained by the fast version of the CBC construction and the ordinary CBC vector z^* do in general not coincide since they have different worst-case errors. Both implementations of the component-by-component construction provide comparable error values, but there is no obvious pattern recognisable which indicates when one of the algorithms provides smaller worst-case errors than the other.

Explanation: To understand the connection between the performance of the CBC algorithm and the decay of the weight sequence γ, we turn back to the worst-case error expression in (5.3). We can rewrite this expression as

$$e_{n,d}^2(z) = \frac{1}{n}\left[\sum_{\emptyset \neq u \subseteq \{1,\dots,d\}} \gamma_u \sum_{k=0}^{n-1} \prod_{j \in u} B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right],$$

where $\gamma_u = \prod_{j \in u} \gamma_j$. Since the weights γ_j form a decreasing sequence, the worst-case error $e_{n,d}(z)$ inherits the weight structure with respect to the coordinates of the original function space. Considering the above expression, this means that the subsets $u \subseteq \{1, ..., d\}$ with large entries contribute less to the value of the sum. Hence, the first coordinates of $z \in Z_n^d$ are more important for the value of $e_{n,d}(z)$ than the latter components. Recalling the component-by-component construction (see Algorithm 1), we see that in each step of the algorithm $e_{n,d}(z_1, \ldots, z_{d-1}, z)$ is minimized over all $z \in Z_n$, where $d \in \{1, \ldots, d_{\max}\}$. If we assume that the decay of the weight sequence γ is fast, then this approach is promising because it minimizes the most important summands first while the factors γ_u of the other summands are so small that these summands are negligible. If, on the other hand, the decay of the γ_j is slow, then the neglected summands are more important. Thus, it is more likely that there are combinations of the z_j which haven't been considered by the CBC algorithm, such that the error value is smaller than $e_{n,d}(z^*)$. This explains the observed numerical results for the different weight sequences.

In the formulation of the component-by-component construction (see Algorithm 1) we search coordinate-wise for the $z \in Z_n$ which minimizes $e_{n,d}^2(z_1, \ldots, z_{d-1}, z)$. But this minimizer is not unique as there could be many z that minimize the worst-case error in each step of the algorithm. In the implementation of the algorithm the minimizer is determined by a simple-minded search that selects the first minimizer z that occurs. Since in the fast version of the CBC algorithm by Nuyens and Cools the search is carried out in a permuted search space, the minimizers may occur in a different order as in the classic CBC algorithm. Moreover, since we worked with double figures which have a rather small mantissa, all numerical values are prone to round-off errors. This behaviour is reflected in our numerical experiments with both versions of the algorithm.

Conclusion: We conclude that under certain conditions, namely fast decaying weights γ_j, the component-by-component construction performs quite well regarding the minimization problem (MP). However, if the weight sequence γ is not behaving well, there is a large number of generating vectors z which provide smaller worst-case errors $e_{n,d}(z)$ than the CBC vector. Therefore, we aim to implement an algorithm which yields better generating vectors than the CBC algorithm and whose performance is especially good when the weights are declining slowly.

5.4 An optimization approach

In order to implement a method which yields generating vectors with smaller worst-case error than the component-by-component construction, we turn back to our initial minimization problem

$$\begin{aligned} &\text{minimize} && e_{n,d}^2(z) \\ &\text{subject to} && z \in Z_n^d = \{0 < \hat{z} < n \,|\, \gcd(\hat{z}, n) = 1\}^d . \end{aligned} \tag{MP}$$

Instead of considering only integer vectors $z \in Z_n^d$, we extend the set of feasible z to generating vectors whose components lie in the interval $[1, n-1]$, i.e. $z \in [1, n-1]^d$. Therefore, we obtain the following relaxed minimization problem

$$\begin{aligned} &\text{minimize} && e_{n,d}^2(z) \\ &\text{subject to} && z \in [1, n-1]^d . \end{aligned} \tag{RMP}$$

Our hope is that it is possible to approach this nonlinear, real-valued optimization problem by means of standard optimization techniques and algorithms, which were not applicable to the integer programming problem (MP) from before. If we are able to find a minimizer $z^* \in [1, n-1]^d$ for the problem (RMP), we intend to search for the nearest integer neighbours $z \in Z_n^d$ of z^* which hopefully also has a small worst-case error.

In this thesis we do only consider the worst-case error $e_{n,d}(z)$ of the weighted unanchored Sobolev space with parameters $\beta_j = 1$ for all $j \in \{1, \dots, d\}$. Here, the squared worst-case error takes the form

$$e_{n,d}^2(z) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right),$$

and the corresponding reproducing kernel Hilbert space has a close connection to the weighted Korobov space with $\alpha = 2$. Since the worst-case error has a similar form for the weighted anchored Sobolev space with different constant sequences $\beta = \{\beta_j\}_{j=1}^d$ and for the weighted Korobov space with different smoothing parameters α, our obtained results are transferable to other weighted RKHS.

To solve the problem (RMP), we interpret $e_{n,d} : [1, n-1]^d \to \mathbb{R}$ as a d-dimensional real-valued function with arguments $z \in [1, n-1]^d$. Then, we apply the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) to the function $e_{n,d}^2$ in order to find a minimizer $z^* \in [1, n-1]^d$ which solves the minimization problem (RMP). The BFGS method is an iterative optimization method which, based on an initial starting point x_0, calculates a sequence of points $\{x_k\}_{k \in \mathbb{N}}$. Under appropriate conditions, this sequence converges superlinearly to a local minimizer of the objective function. The pseudocode of the BFGS algorithm is given below.

Algorithm 3 BFGS method

Given starting point x_0, convergence tolerance $\varepsilon > 0$ and inverse Hessian approximation H_0 :
Set $k = 0$

while $\|\nabla f(x_k)\| > \varepsilon$ **do**

1. Compute search direction via $p_k = -H_k \nabla f(x_k)$
2. Perform a line search to obtain the step size α_k (Wolfe-Powell condition)
3. Set $x_{k+1} = x_k + \alpha_k p_k$
4. Set $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $\rho_k = \frac{1}{y_k^T s_k}$
5. Compute the inverse Hessian approximation H_{k+1} via

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

6. $k = k + 1$

end while

In essence, the algorithm approximates the inverse of the Hessian matrix $\nabla^2 f$ of the objective function f and then performs an iterative procedure that is based on Newton's method. We further

note that the BFGS algorithm is theoretically designed for unconstrained optimization problems, our problem, however, is restricted to the d-dimensional cube $[1, n-1]^d$. Nevertheless, we will ignore this fact for a moment and still apply the BFGS method to our problem.

The formulation of the algorithm shows that the gradient of the objective function $e^2_{n,d}$ has to exist in order to apply the BFGS method. Examining the worst-case error expression

$$e^2_{n,d}(z) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right) ,$$

we realize that $e^2_{n,d}$ is not continuously differentiable with respect to z since the fractional part $\{\cdot\}$ inside $B_2(\cdot)$ destroys the continuity of the partial derivatives. Nevertheless, we can erase the points of discontinuity by smoothing the Bernoulli polynomial $B_2(x) = x^2 - x + \frac{1}{6}$ in the following sense:

For fixed $n \in \mathbb{N}$ and $1 \leq k \leq n-1$ the following figure displays the behaviour of the function $f(z) := B_2\left(\left\{\frac{k \cdot z}{n}\right\}\right)$, where $z \in [1, n-1]$. In the figure we used the values $n = 23$ and $k = 5$.

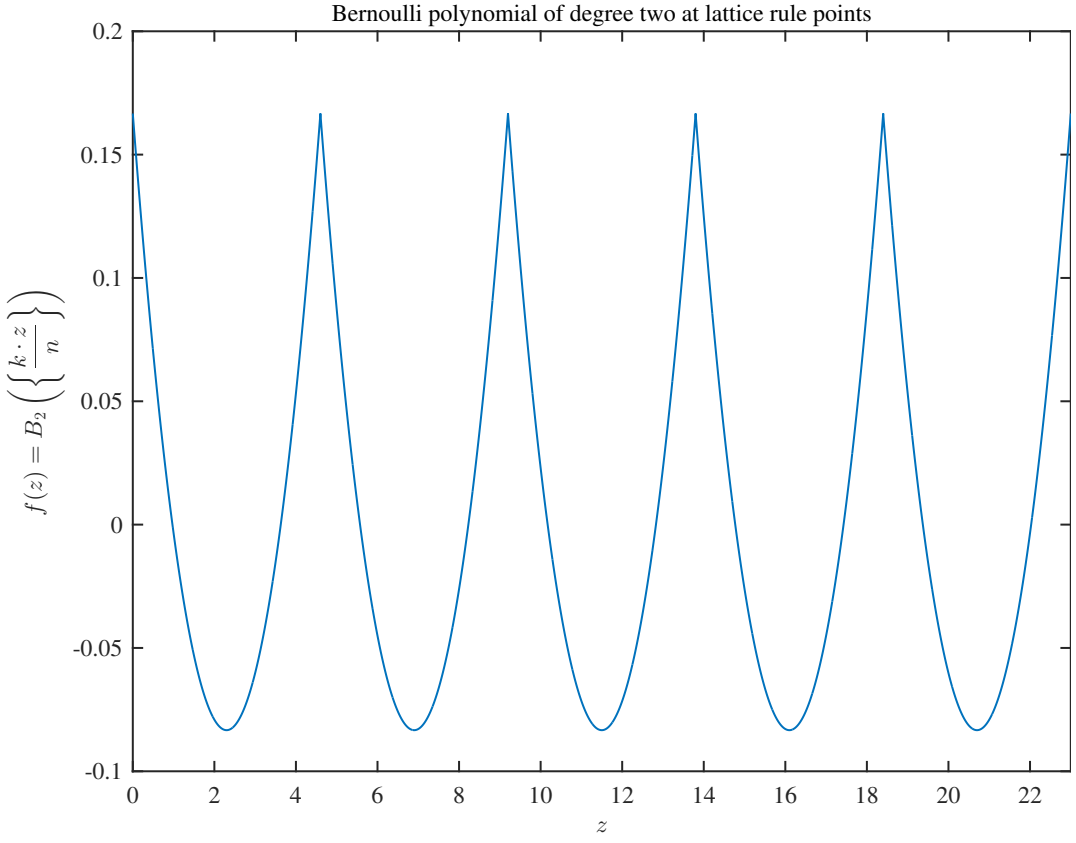

Figure 1: Illustration of the behaviour of $B_2\left(\left\{\frac{k \cdot z}{n}\right\}\right)$

Since the corresponding derivative with respect to z is given by

$$\frac{\partial}{\partial z} B_2\left(\left\{\frac{k\cdot z}{n}\right\}\right) = \frac{k}{n}\cdot B_2'\left(\left\{\frac{k\cdot z}{n}\right\}\right) = \frac{2k}{n}\left(\left\{\frac{k\cdot z}{n}\right\} - \frac{1}{2}\right),$$

the function $f(z) := B_2\left(\left\{\frac{k\cdot z}{n}\right\}\right)$ is discontinuous in the points $z_k = \frac{l\cdot n}{k}$ with $l \in \{1,\ldots,n-1\}$. In order to replace $B_2(x)$ by a function $\widetilde{B}_2(x)$ such that the expression $\widetilde{B}_2\left(\left\{\frac{k\cdot z}{n}\right\}\right)$ has smooth transitions between the various parabolic arcs, we consider the following situation:

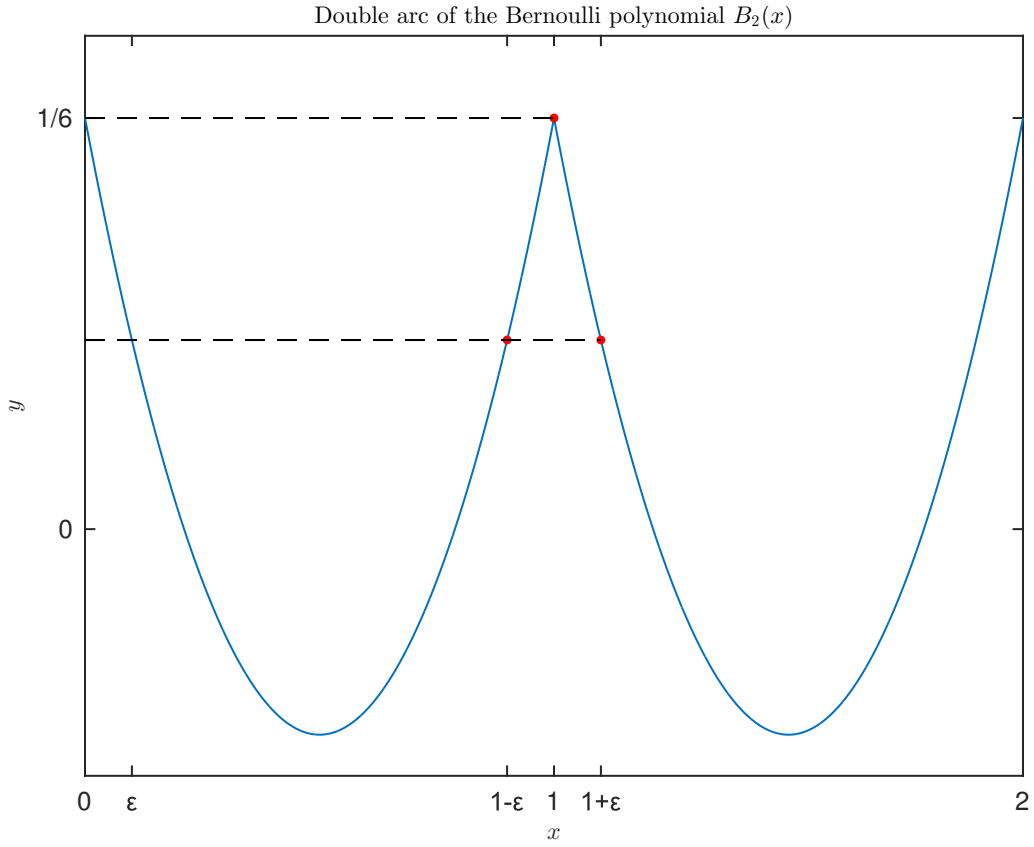

Figure 2: Illustration of two adjoining arcs of the function $B_2(x)$

We want to perform a smooth polynomial fit through the three marked points in Figure 2 above, therefore we use a polynomial p of degree 4 of the form $p(x) = c_1x^4 + c_2x^3 + c_3x^2 + c_4x + c_5$, where $c_1,\ldots,c_5 \in \mathbb{R}$. Thus, for $\varepsilon > 0$ small, p must satisfy the following conditions:

(1) $p(1-\varepsilon) = B_2(1-\varepsilon) = B_2(\varepsilon) = \varepsilon^2 - \varepsilon + \frac{1}{6}$ (2) $p'(1-\varepsilon) = B_2'(1-\varepsilon) = 2(1-\varepsilon) - 1 = 1 - 2\varepsilon$

(3) $p(1+\varepsilon) = B_2(1-\varepsilon) = B_2(\varepsilon) = \varepsilon^2 - \varepsilon + \frac{1}{6}$ (4) $p'(1+\varepsilon) = B_2'(\varepsilon) = 2\varepsilon - 1$

(5) $p(1) = B_2(1) = B_2(0) = \frac{1}{6}$.

These five conditions can be reformulated as the system of linear equations $Ac = b$, where

$$A = \begin{pmatrix} (1-\varepsilon)^4 & (1-\varepsilon)^3 & (1-\varepsilon)^2 & (1-\varepsilon) & 1 \\ (1+\varepsilon)^4 & (1+\varepsilon)^3 & (1+\varepsilon)^2 & (1+\varepsilon) & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 4(1-\varepsilon)^3 & 3(1-\varepsilon)^2 & 2(1-\varepsilon) & 1 & 0 \\ 4(1+\varepsilon)^3 & 3(1+\varepsilon)^2 & 2(1+\varepsilon) & 1 & 0 \end{pmatrix} \text{ and } b = \begin{pmatrix} \varepsilon^2 - \varepsilon + \frac{1}{6} \\ \varepsilon^2 - \varepsilon + \frac{1}{6} \\ \frac{1}{6} \\ 1 - 2\varepsilon \\ 2\varepsilon - 1 \end{pmatrix}.$$

Solving this system of linear equations, we obtain the vector $c = (c_1, \ldots, c_5)^T$ which consists of the coefficients $c_1, \ldots, c_5$ of the polynomial $p(x) = c_1x^4 + c_2x^3 + c_3x^2 + c_4x + c_5$. Using the symmetry of B_2, one can use this polynomial to define the function $\widetilde{B}_2(x)$ for $0 \le x \le 1$ as

$$\widetilde{B}_2(x) := \begin{cases} p(x+1) & \text{for } x < \varepsilon\,, \\ x^2 - x + \frac{1}{6} & \text{for } x \in [\varepsilon, 1-\varepsilon]\,, \\ p(x) & \text{for } x > 1-\varepsilon\,. \end{cases}$$

The following graph shows that the function $\widetilde{B}_2(x)$ achieves the desired smoothing for the function $\widetilde{B}_2\left(\left\{\frac{k\cdot z}{n}\right\}\right)$. Here, we illustrated the smoothed function $\widetilde{B}_2(x)$ in the same setting as for Figure 1.

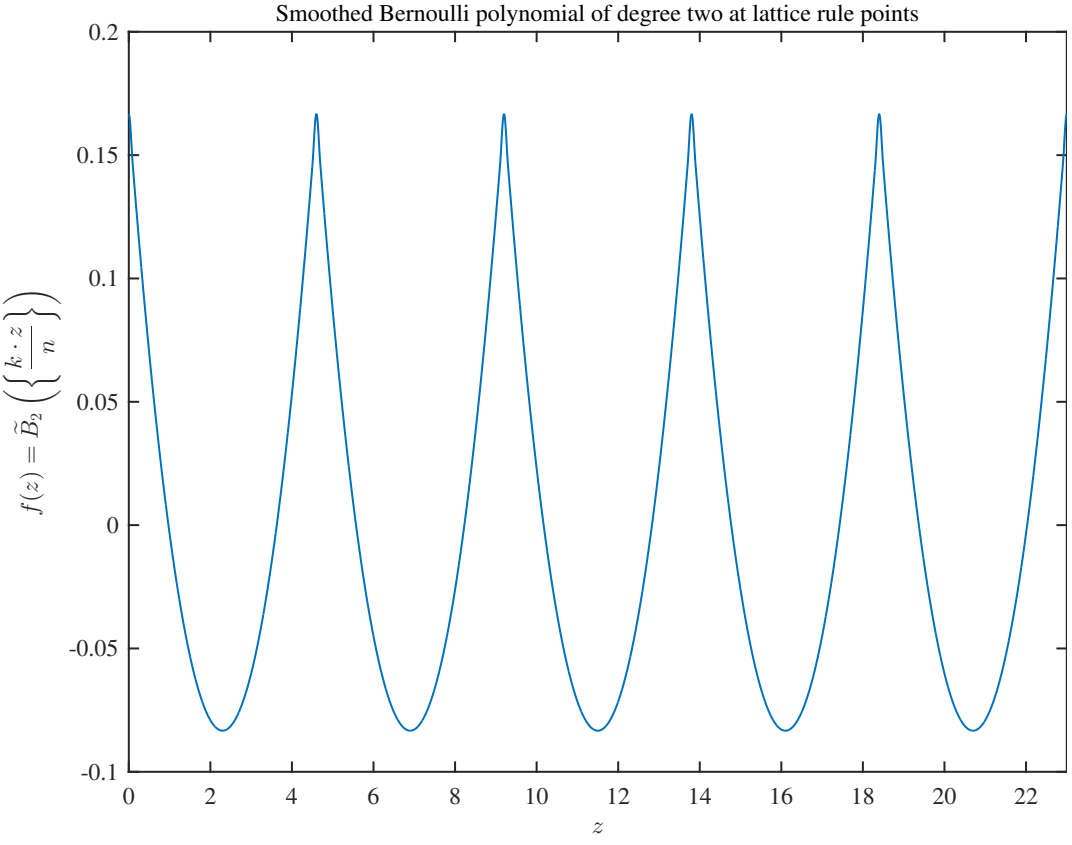

Figure 3: Illustration of the behaviour of $\widetilde{B}_2\left(\left\{\frac{k\cdot z}{n}\right\}\right)$

Using the newly defined $\widetilde{B}_2(\cdot)$ above, we are able to apply the BFGS method to our function $e^2_{n,d}(z)$, where we replaced the Bernoulli polynomial of degree two by $\widetilde{B}_2$. The corresponding gradient is given by $\nabla e^2_{n,d}(z) = (\frac{\partial e^2_{n,d}}{\partial z_1}, \ldots, \frac{\partial e^2_{n,d}}{\partial z_d})^T$ with partial derivatives of the form

$$\begin{aligned}
\frac{\partial e^2_{n,d}}{\partial z_i} &= \frac{1}{n}\sum_{k=0}^{n-1} \frac{\partial}{\partial z_i} \prod_{j=1}^{d} \left(1 + \gamma_j \cdot \widetilde{B}_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right) \\
&= \frac{1}{n}\sum_{k=0}^{n-1} \prod_{\substack{j=1 \\ j\neq i}}^{d} \left(1 + \gamma_j \cdot \widetilde{B}_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right) \cdot \left(\gamma_j \cdot \frac{\partial}{\partial z_i} \widetilde{B}_2\left(\left\{\frac{k \cdot z_i}{n}\right\}\right)\right) \\
&= \frac{1}{n}\sum_{k=0}^{n-1} \prod_{\substack{j=1 \\ j\neq i}}^{d} \left(1 + \gamma_j \cdot \widetilde{B}_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right) \cdot \left(\frac{\gamma_j \cdot k}{n} \cdot \widetilde{B}_2'\left(\left\{\frac{k \cdot z_i}{n}\right\}\right)\right) .
\end{aligned}$$

The associated derivative of $\widetilde{B}_2(x)$ is further given as

$$\widetilde{B}_2'(x) := \begin{cases} p'(x+1) & \text{for } x < \varepsilon\,, \\ 2x - 1 & \text{for } x \in [\varepsilon, 1-\varepsilon]\,, \\ p'(x) & \text{for } x > 1-\varepsilon\,. \end{cases}$$

As it is not clear how to choose the starting point $z_0 \in [1, n-1]^d$, we decide to select a number of random vectors whose components are uniformly distributed on Z_n and then apply Algorithm 3 to them.

Table 6: $d = 4, n = 101, \gamma_j = 1/j^2$

number	initial vector z_0	BFGS vector z^*
1	$(43, 44, 27, 29)$	$(43.0332, 43.9968, 26.9875, 28.9661)$
2	$(29, 50, 24, 43)$	$(28.9693, 50.1183, 24.0076, 43.0333)$
3	$(47, 1, 41, 37)$	$(47.0772,\ 0.7546, 40.9972, 36.9987)$
4	$(35, 44, 12, 30)$	$(34.9829, 44.0003, 11.9214, 30.0042)$
5	$(30, 31, 25, 13)$	$(30.0031, 31.0066, 25.0525, 12.9685)$
6	$(41, 50, 46, 34)$	$(40.9716, 50.1187, 45.9913, 33.9234)$

The numerical experiments show that for all used starting points z_0 the procedure returns points z^* which are very close to the initial points z_0. Similar results were also obtained for different weights γ_j and different values of n and d. This behaviour indicates that the function $e^2_{n,d}(\cdot)$ possesses a large number of local minima with respect to the used BFGS method. Therefore, it is not possible to find a global solution for the minimization problem (RMP) with our approach since the method is immediately stuck in one of the local minima near z_0.

In order to visualize why our optimization attempt has failed, we illustrate the function $e^2_{n,d}$ with respect to the first two coordinates z_1, z_2 in a 3-dimensional plot. For $d = 4, n = 53$ and $z_3, z_4 \in Z_n$ fixed, we obtain the following figures:

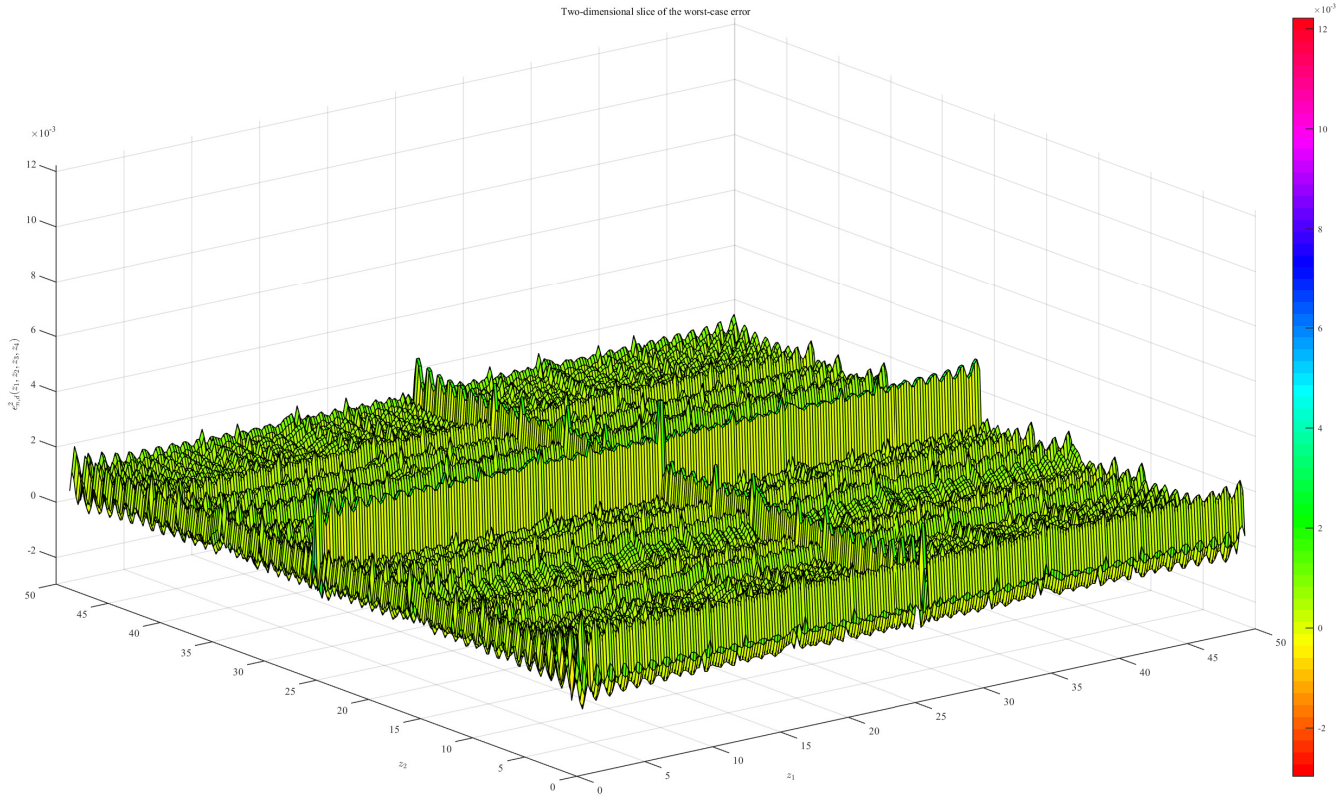

Figure 4: Two-dimensional slice of the squared worst-case error $e^2_{n,d}(z_1, \ldots, z_4)$

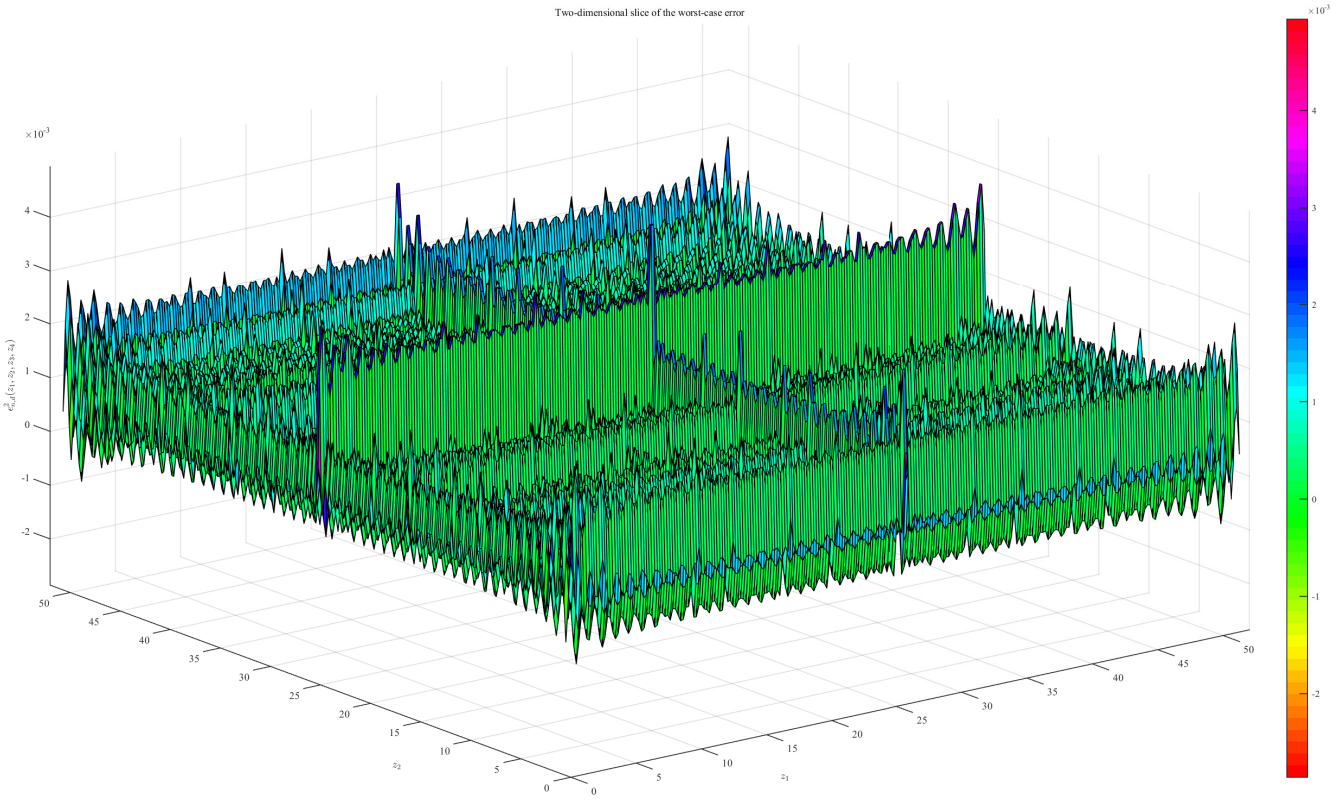

Figure 5: Two-dimensional slice of the squared worst-case error $e^2_{n,d}(z_1, \ldots, z_4)$

The plot depicts the behaviour of the objective function $e^2_{n,d}$ and reveals that the function possesses a plethora of local minima. It also explains why our optimization attempt was foredoomed to fail, since for such a chaotic function the minimization via standard methods is impossible. The application of the BFGS algorithm for some initial vector z_0 to our function resulted in the fact that the point sequence $\{z_k\}_{k\in\mathbb{N}}$ generated by the BFGS method converged to some of the local minima in the direct neighbourhood of z_0. The next figure further illustrates this fact, here we plotted the one-dimensional function $e^2_{n,d}(z_1, z_2, z_3, z_4)$ obtained by keeping the components z_2, z_3, z_4 fixed and ranging z_1 over the set $[0, n-1]$.

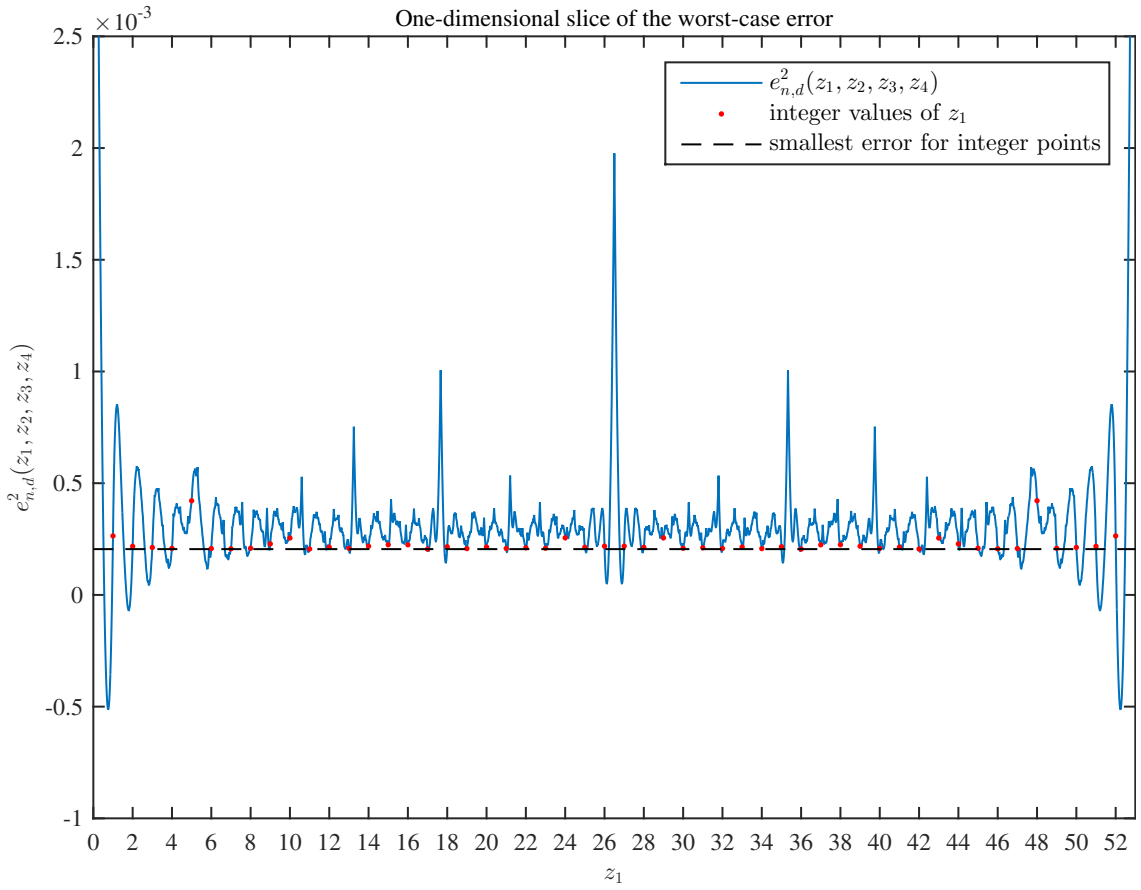

Figure 6: One-dimensional slice of the squared worst-case error $e^2_{n,d}(z_1, \ldots, z_4)$

This one-dimensional plot shows again that $e^2_{n,d}$ has a large number of local minima. Further, we note that the minimal values of $e^2_{n,d}(z^*, z_2, z_3, z_4)$ with real-valued $z^* \in [1, n-1]$ have no connection to the worst-case error $e^2_{n,d}(z, z_2, z_3, z_4)$ of the nearest integer neighbour $z \in Z^d_n$. We see that for $z^* \in [1, n-1]$ with a small squared worst-case error $e^2_{n,d}(z^*, z_2, z_3, z_4)$, the nearest integer neighbour does not necessarily also have a small squared worst-case error compared to other integer vectors (z, z_2, z_3, z_4) with $z \in Z_n$. This means that even if we were able to find the global minimizer z^* of

$e^2_{n,d}$ in the set $[1, n-1]$, there would be no guarantee that the associated nearest integer neighbour also has a small error value. Our results show that a standard optimization approach is not possible in our setting as the objective function $e^2_{n,d}$ is too ill-behaved. As a result we turn back to algorithms which are similar to the component-by-component construction.

6 Branching in the component-by-component algorithm

6.1 The occurrence of multiple CBC vectors

In this chapter we will examine the occurrence of different generating vectors for the two versions of the component-by-component algorithm which we observed in our numerical experiments in the previous chapter. Firstly, we introduce the following notion.

Definition 6.1 (Branch vector)
Let n be a prime number, $d \in \mathbb{N}$ the dimension and $\gamma = \{\gamma_j\}_{j=1}^d$ a sequence of positive weights. Then two integer vectors $z \neq \bar{z} \in Z_n^d$ obtained by the component-by-component construction (Algorithm 1) in a weighted RKHS with associated weight sequence γ are called related branch vectors if there exists a number $s \in \mathbb{N}_{\geq 2}$ such that

(i) $z_j = \bar{z}_j$ *for* $j = 1, \ldots, s-1$ *and* $z_s \neq \bar{z}_s$

(ii) $e_{n,s}(z) = e_{n,s}(\bar{z})$.

The occurrence of multiple minima and minimizers in the component-by-component algorithm will be called "branching". This name arises from the fact that if we have multiple minimizers in a certain step of the algorithm, then all resulting generating vectors are per se CBC vectors and so we obtain various branches. The development of the components $z_{s+1}, \ldots, z_d$ of a CBC vector for which branching occurred in step s can be completely different for the various vectors. Therefore, the associated worst-case error $e_{n,d}(z)$ can also vary for the different branches received by the component-by-component construction.

Remark: In each minimization step of the component-by-component construction the value of z_d with $d \in \{1, \ldots, d_{\max}\}$ is only influenced by $\gamma_1, \ldots, \gamma_{d-1}$. Thus, the CBC vector of dimension d only depends on the weights $\gamma_1, \ldots, \gamma_{d-1}$. The worst-case error $e_{n,d}(z)$, however, is determined by all weights $\gamma_1, \ldots, \gamma_d$. We can prove this behaviour by rewriting the worst-case error as

$$\begin{aligned} e_{n,d}^2(z) &= -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) \\ &= -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} c_k \cdot \left(\beta_d + \gamma_d \cdot \omega \left(\left\{ \frac{k \cdot z_d}{n} \right\} \right) \right) \\ &= \underbrace{-\prod_{j=1}^{d} \beta_j + \frac{\beta_d}{n} \sum_{k=0}^{n-1} c_k}_{\text{constant}} + \frac{\gamma_d}{n} \sum_{k=0}^{n-1} c_k \cdot \omega \left(\left\{ \frac{k \cdot z_d}{n} \right\} \right) , \end{aligned}$$

where $c_k := \prod_{j=1}^{d-1} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right)$. In the minimization step we only have to find $z_d \in Z_n$ such that the last sum in minimized and this sum is independent of γ_d and only depends on the c_k which depend on $\gamma_1, \ldots, \gamma_{d-1}$. Thus, the minimization is not influenced by γ_d.

The next theorem shows that branching is not unlikely to appear, it actually always occurs.

Theorem 6.2 (Branching in the second step)
Let n be a prime number and $\gamma = \{\gamma_1, \ldots, \gamma_d\}$ a weight sequence of positive weights. If $z^* \in Z_n^d$ is a generating vector of a rank-one lattice rule obtained by Algorithm 1 with worst-case error

$$e_{n,d}^2(z) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) ,$$

then there exists another CBC vector $\bar{z} \in Z_n^d$ such that $e_{n,2}(z^*) = e_{n,2}(\bar{z})$. Furthermore, the second component of $\bar{z}$ is given by $\bar{z}_2 = (z_2^*)^{-1}$, where $(z_2^*)^{-1}$ is the multiplicative inverse of z_2^* modulo n.

Proof. The squared worst-case error $e_{n,d}^2(z)$ is given by

$$e_{n,d}^2(z) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) ,$$

where n is prime and $z_j \in \{1, \ldots, n-1\}$ for $j = 1, \ldots, d$. We want to show that branching always occurs in the second step:

Let $z^* := (z_1^*, z_2^*) = (1, z_2^*)$ be the CBC vector obtained by minimizing the second component of z^*. First, we can rewrite the error term $e_{n,2}^2(z)$ as follows

$$\begin{aligned}
e_{n,2}^2(z) &= -\beta_1 \beta_2 + \frac{1}{n} \sum_{k=0}^{n-1} \left(\beta_1 + \gamma_1 \cdot \omega \left(\left\{ \frac{k \cdot z_1}{n} \right\} \right) \right) \left(\beta_2 + \gamma_2 \cdot \omega \left(\left\{ \frac{k \cdot z_2}{n} \right\} \right) \right) \\
&= -\beta_1 \beta_2 + \frac{1}{n} \sum_{k=0}^{n-1} \left[\beta_1 \beta_2 + \gamma_1 \beta_2 \cdot \omega \left(\left\{ \frac{k \cdot z_1}{n} \right\} \right) + \gamma_2 \beta_1 \cdot \omega \left(\left\{ \frac{k \cdot z_2}{n} \right\} \right) + \right. \\
&\quad \left. + \gamma_1 \gamma_2 \beta_1 \beta_2 \cdot \omega \left(\left\{ \frac{k \cdot z_1}{n} \right\} \right) \omega \left(\left\{ \frac{k \cdot z_2}{n} \right\} \right) \right] \\
&= \underbrace{-\beta_1 \beta_2 + \frac{1}{n} \sum_{k=0}^{n-1} \beta_1 \beta_2}_{=0} + \underbrace{\frac{\gamma_1 \beta_2}{n} \sum_{k=0}^{n-1} \omega \left(\left\{ \frac{k \cdot z_1}{n} \right\} \right)}_{\text{constant}} + \underbrace{\frac{\gamma_2 \beta_1}{n} \sum_{k=0}^{n-1} \omega \left(\left\{ \frac{k \cdot z_2}{n} \right\} \right)}_{\text{constant}} + \\
&\quad + \gamma_1 \gamma_2 \beta_1 \beta_2 \sum_{k=0}^{n-1} \omega \left(\left\{ \frac{k \cdot z_1}{n} \right\} \right) \omega \left(\left\{ \frac{k \cdot z_2}{n} \right\} \right) .
\end{aligned}$$

Since n is prime the term $\left\{ \frac{k \cdot z_j}{n} \right\}$ runs through all fractions of the form $\frac{l}{n}$ with $l \in \{0, \ldots, n-1\}$ with the order determined by z_j. Thus, everything in the error expression is constant except for the last term. Hence, if we can find another generating vector $\bar{z} = (1, \bar{z}_2)$ such that the last sum term has the same value as for z^*, then we get that also $e_{n,2}^2(z^*) = e_{n,2}^2(\bar{z})$.

Our candidate for $\bar{z}$ is given by $\bar{z} := (1, (z_2^*)^{-1})$, where $(z_2^*)^{-1}$ is the multiplicative inverse of z_2^* modulo n, i.e. in the finite field $\mathbb{F}_n = \mathbb{Z}_n$. For simplicity's sake, we denote $\omega\left(\left\{\frac{a}{n}\right\}\right)$ by $\omega(a)$ with $a \in \mathbb{Z}$. For the above $\bar{z}$ we obtain the following:

(1) Consider $e_{n,2}^2(z^*)$ and in particular the last sum given as

$$\sum_{k=0}^{n-1} \omega\left(\left\{\frac{k}{n}\right\}\right) \omega\left(\left\{\frac{k \cdot z_2^*}{n}\right\}\right) \triangleq \sum_{k=0}^{n-1} \omega(k) \cdot \omega(\underbrace{k \cdot z_2^*}_{\in \mathbb{F}_n}) .$$

We can regroup the summation by taking $k = \bar{z}_2 + a$, where a runs from 0 to $n-1$

$$\omega(\bar{z}_2 + a) \cdot \omega(\underbrace{(\bar{z}_2 + a) \cdot z_2^*}_{= \bar{z}_2 \cdot z_2^* + a \cdot z_2^* = 1 + a \cdot z_2^*}) = \omega(\bar{z}_2 + a) \cdot \omega(1 + a \cdot z_2^*) .$$

Since for the maps $\varphi : \mathbb{F}_n \to \mathbb{F}_n$, $a \mapsto \bar{z}_2 + a$ and $\psi : \mathbb{F}_n \to \mathbb{F}_n, a \mapsto 1 + a \cdot z_2^*$ we have that $\mathrm{im}(\varphi) = \mathrm{im}(\psi) = \mathbb{F}_n$ and φ, ψ are in particular bijective, we obtain all values $0, \ldots, n-1$ by running a from 0 to $n-1$. Hence we have

$$\sum_{a=0}^{n-1} \omega(\bar{z}_2 + a) \cdot \omega(1 + a \cdot z_2^*) = \sum_{a=0}^{n-1} \omega(\bar{z}_2 + a) \cdot \omega((\bar{z}_2 + a) \cdot z_2^*) = \sum_{k=0}^{n-1} \omega(k) \cdot \omega(k \cdot z_2^*) .$$

(2) Similarly we consider $e_{n,2}^2(\bar{z}_2)$ and obtain for k running like $k = 1 + a \cdot z_2^*$

$$\sum_{k=0}^{n-1} \omega(k) \cdot \omega(k \cdot \bar{z}_2) = \sum_{a=0}^{n-1} \omega(1 + a \cdot z_2^*) \cdot \omega((1 + a \cdot z_2^*) \cdot \bar{z}_2) = \sum_{a=0}^{n-1} \omega(1 + a \cdot z_2^*) \cdot \omega(\bar{z}_2 + a) .$$

We see that both resulting sums are equal and so we obtain according to our analysis above

$$e_{n,2}(z^*) = e_{n,2}(\bar{z}) .$$

■

Remark: Note that the above theorem holds for a general kernel function ω since we do not require any additional properties of the function ω in the proof. For a symmetric kernel function ω we can furthermore restrict the set of feasible generating vectors z to Z_m^d instead of Z_n^d, where $m := (n-1)/2$. This restriction makes the definition of a branch vector unambiguous because it excludes the existence of branch vectors of the form $z = (z_1, \ldots, z_{s-1}, z_s, z_{s+1}, \ldots, z_d)$ and $\bar{z} = (z_1, \ldots, z_{s-1}, n - z_s, \bar{z}_{s+1}, \ldots, \bar{z}_d)$ for which trivially $e_{n,s}(z) = e_{n,s}(\bar{z})$. Additionally, we notice that if the multiplicative inverse of z_2^* coincides with z_2^* itself, then there is no branching in the second step as promised by Theorem 6.2.

Thus, branching always occurs in the CBC algorithm. Since all branch vectors are constructed by the CBC algorithm, the corresponding worst-case errors still obey the error bounds as in the Theorems 5.3 and 5.4 by Kuo. However, it is possible that there are notable differences between the worst-case errors of different branch vectors. In the following we display the results of some numerical experiments in the weighted unanchored Sobolev space which illustrate these differences.

Table 7: $d = 20, \beta_j = 1, \gamma_j = (0.8)^j$

n	$e_{n,d}(z_1)$	$e_{n,d}(z_2)$	relative difference in %
101	3.0220e-02	3.0569e-02	1.1536
139	2.3592e-02	2.3897e-02	1.2907
263	1.4549e-02	1.4877e-02	2.2549
349	1.1566e-02	1.1857e-02	2.5112
563	8.1135e-03	8.2462e-03	1.6344
677	7.0436e-03	7.1968e-03	2.1751
859	5.8505e-03	5.9689e-03	2.0234
947	5.3661e-03	5.4921e-03	2.3496

Table 8: $d = 20, \beta_j = 1, \gamma_j = 1/j^{\frac{3}{2}}$

n	$e_{n,d}(z_1)$	$e_{n,d}(z_2)$	relative difference in %
409	4.0540e-03	4.1214e-03	1.6615
593	2.9123e-03	2.9693e-03	1.9566
677	2.6241e-03	2.6805e-03	2.1478
821	2.2378e-03	2.2845e-03	2.0855
827	2.2461e-03	2.2841e-03	1.6921
907	2.0311e-03	2.0840e-03	2.6032
967	1.9215e-03	1.9586e-03	1.9318
991	1.8786e-03	1.9131e-03	1.8357

Note that the differences in the error values strongly depend on the weight sequence γ and the number of quadrature points n of the respective lattice rules. The next algorithm constructs and collects all possible branches of generating vectors that occur in the specific component-by-component construction. Here, we used the fast version of the CBC algorithm by Nuyens and Cools (see [5]) as point of origin.

Remark: The algorithm below determines all possible branch vectors even though their number may grow exponentially in d. However, this behaviour hasn't been observed in our numerical experiments so far. In fact, we only observed branching in the second step as we proved theoretically. It remains to investigate whether it is possible to deliberately determine the weights γ_j such that branching occurs in multiple steps.

Listing 1: Fast version of the Branch-algorithm for the CBC construction

```
% Author: Adrian Ebert
% Based on the fast CBC algorithm by Dirk Nuyens
%
% function [z, e2] = FCBC_Branch(n, s_max, omega, gamma, beta)
%
% inputs
%   n           number of quadrature points, integer, scalar
%   s_max       dimension, integer, scalar
%   omega       function handle for kernel function \omega
%   gamma       weight sequence, real, 1 x s_max vector
%   beta        constant sequence, real, 1 x s_max vector
%
% outputs
%   z           generating vectors of the lattice rules, s_max x l vector
%   e2          squared worst-case errors per dimension, s_max x l vector

function [z, e2] = FCBC_Branch(n, s_max, omega, gamma, beta)

ep = 1e-10; %may need adjustment depending on the magnitude of the error

if ~isprime(n), error('n must be prime'); end;
z = 0; m = (n-1)/2;
g = generatorp(n);
perm = zeros(m, 1);
perm(1) = 1; for j=1:m-1, perm(j+1) = mod(perm(j)*g, n); end;
perm = min(n - perm, perm);
psi = omega(perm/n);
fft_psi = fft(psi);
q = ones(m, 1);

for s = 1:s_max
    for i=1:size(z,2)

        E2 = ifft(fft_psi .* fft(q(:,i)));
        E2 = real(E2);
        [min_E2,w] = sort(E2);
        h = min_E2 - min_E2(1)*ones(m,1); h = h(h<ep);
        w = w(1:length(h));
        z2 = [z(:,i);0];

        if s == 1, w = 1; z_new = 1;

        else
            for p=2:length(w), z2 = [z2,[z(:,i);0]]; end;
            for j=1:length(w), z2(s,j) = perm(w(j)); end;

            if i == 1, z_new = z2;
            else z_new = [z_new,z2]; end
        end

        q2 = (beta(s) + gamma(s) * psi([w(1):-1:1 m:-1:w(1)+1])) .* q(:,i);

        if s == 1, q_new = q2;
```

```
        else
            for j=2:length(w), q2=[q2,(beta(s) + gamma(s) * ...
                    psi([w(j):-1:1 m:-1:w(j)+1])) .* q(:,i)]; end;

            if i==1, q_new = q2;
            else q_new = [q_new,q2]; end
        end
    end
    z = z_new; q = q_new;
end

e2 = zeros(1,size(z,2));

for i=1:size(z,2), e2(i) = wce(n, z(:,i)', omega, gamma, beta); end;
end
```

6.2 Conditions on the weights $\gamma = \{\gamma_j\}_{j=1}^d$

Furthermore, we are interested in examining conditions for the weights $\gamma = \{\gamma_j\}_{j=1}^d$ which assure that a particular branch is the best possible CBC vector with respect to the worst-case error without having to calculate all possible branches. In order to derive these conditions, we consider the weighted unanchored Sobolev space with parameters $\beta_j = 1$ for $j = 1, \ldots, d$ in which the worst-case error takes the form

$$e_{n,d}^2(z) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(1 + \gamma_j B_2\left(\left\{\frac{k \cdot z}{n}\right\}\right)\right).$$

Now, we assume that branching occurs in step s of the component-by-component construction with $1 < s < d$ and we aim to select the best CBC vector amongst the different branches. This means there exist CBC vectors that have been constructed until $z = (z_1, \ldots, z_{s-1})$ such that there are multiple $z_s \in Z_m$ that yield the same value $e_{n,s}^2(z_1, \ldots, z_s)$. Next, we are looking at stage $s+1$ and intend to select a unique optimal branch, that is choose z_s^B and $z_{s+1}^B(z_s^B)$ such that

$$e_{n,s+1}^2(z_1, \ldots, z_s^B, z_{s+1}^B(z_s^B)) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{s-1} \underbrace{\left(1 + \gamma_j B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right)}_{=: p_{d-1}(k)} \cdot \left(1 + \gamma_s B_2\left(\left\{\frac{k \cdot z_s^B}{n}\right\}\right)\right) \cdot \left(1 + \gamma_{s+1} B_2\left(\left\{\frac{k \cdot z_{s+1}^B(z_s^B)}{n}\right\}\right)\right)$$

is strictly smaller than the worst-case error

$$e_{n,s+1}^2(z_1, \ldots, z_s^A, z_{s+1}^A(z_s^A)) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{s-1} \underbrace{\left(1 + \gamma_j B_2\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right)}_{=: p_{d-1}(k)} \cdot \left(1 + \gamma_s B_2\left(\left\{\frac{k \cdot z_s^A}{n}\right\}\right)\right) \cdot \left(1 + \gamma_{s+1} B_2\left(\left\{\frac{k \cdot z_{s+1}^A(z_s^A)}{n}\right\}\right)\right)$$

of any alternative choice $(z_s^A, z_{s+1}^A(z_s^A)) \neq (z_s^B, z_{s+1}^B(z_s^B))$. Now suppose the worst choice of $z_{s+2}, \ldots, z_d$ for the optimal branch is better than the optimal choice of $z_{s+2}, \ldots, z_d$ in any alternative branch. Estimating the range of $B_2(x)$, this would imply

$$\begin{aligned} & -1 + \frac{1}{n} \sum_{k=0}^{n-1} p_{d-1}(k) \left(1 + \gamma_s B_2\left(\left\{\frac{k \cdot z_s^B}{n}\right\}\right)\right) \left(1 + \gamma_{s+1} B_2\left(\left\{\frac{k \cdot z_{s+1}^B}{n}\right\}\right)\right) \prod_{j=s+2}^{d} \left(1 + \frac{\gamma_j}{6}\right) \\ \leq \; & -1 + \frac{1}{n} \sum_{k=0}^{n-1} p_{d-1}(k) \left(1 + \gamma_s B_2\left(\left\{\frac{k \cdot z_s^A}{n}\right\}\right)\right) \left(1 + \gamma_{s+1} B_2\left(\left\{\frac{k \cdot z_{s+1}^A}{n}\right\}\right)\right) \prod_{j=s+2}^{d} \left(1 - \frac{\gamma_j}{12}\right), \end{aligned}$$

since the minimal value of $B_2(x) = x^2 - x + \frac{1}{6}$ is $-\frac{1}{12}$ and the maximal value of $B_2(x)$ is $\frac{1}{6}$ attained

at $x = 0$. Adding one on both sides and rearranging the inequality yields

$$\prod_{j=s+2}^{d} \frac{\left(1-\frac{\gamma_j}{12}\right)}{\left(1+\frac{\gamma_j}{6}\right)} \geq \frac{\overbrace{\sum_{k=0}^{n-1} p_{d-1}(k)\left(1+\gamma_s B_2\left(\left\{\frac{k\cdot z_s^B}{n}\right\}\right)\right)\left(1+\gamma_{s+1}B_2\left(\left\{\frac{k\cdot z_{s+1}^B}{n}\right\}\right)\right)}^{=:\text{(B)}}}{\underbrace{\sum_{k=0}^{n-1} p_{d-1}(k)\left(1+\gamma_s B_2\left(\left\{\frac{k\cdot z_s^A}{n}\right\}\right)\right)\left(1+\gamma_{s+1}B_2\left(\left\{\frac{k\cdot z_{s+1}^A}{n}\right\}\right)\right)}_{=:\text{(A)}}}.$$

Considering the Bernoulli polynomial of degree two $B_2(x) = x^2 - x + \frac{1}{6}$ at a point $x = \frac{l}{n}$ with $l \in \{1, \ldots, n-1\}$, we obtain that

$$B_2\left(\frac{l}{n}\right) = \frac{l^2}{n^2} - \frac{l}{n} + \frac{1}{6} = \frac{6l^2 - 6ln + n^2}{6n^2} = \frac{c}{6n^2} \in \left[-\frac{1}{12}, \frac{1}{6}\right]$$

for some $c \in \left[-\frac{n^2}{2}, n^2\right] \cap \mathbb{Z}$. In addition, we assume that we have rational weights $\gamma_j = \frac{q_j}{M_j}$ with $q_j, M_j \in \mathbb{N}$ and without loss of generality $q_j \leq M_j$. Then every factor of the form

$$\left(1+\gamma_j B_2\left(\left\{\frac{k\cdot z_j}{n}\right\}\right)\right),$$

where $k \in \{0, 1, \ldots, n-1\}$, can be written as

$$\left(1+\frac{q_j}{M_j}\cdot\frac{l}{6n^2}\right) = \frac{r_j}{M_j\cdot 6n^2}$$

for some natural number $r_j \in \mathbb{N}$. Hence, we can rewrite the term (B) as

$$\begin{aligned}
\text{(B)} &= \sum_{k=0}^{n-1} p_{d-1}(k)\left(1+\gamma_s B_2\left(\left\{\frac{k\cdot z_s^B}{n}\right\}\right)\right)\left(1+\gamma_{s+1}B_2\left(\left\{\frac{k\cdot z_{s+1}^B}{n}\right\}\right)\right) \\
&= \sum_{k=0}^{n-1}\prod_{j=1}^{s+1}\frac{r_j(k)}{M_j\cdot 6n^2} = \sum_{k=0}^{n-1}\frac{R(k)}{\prod_{j=1}^{s+1} M_j\cdot(6n^2)^{s+1}} = \frac{q^*}{\prod_{j=1}^{s+1} M_j\cdot(6n^2)^{s+1}},
\end{aligned}$$

for some $q^* \in \mathbb{N}$ (similar for $\textcircled{A}$). Since by assumption $\textcircled{B} \leq \textcircled{A}$, we always have

$$\begin{aligned}
\frac{\textcircled{B}}{\textcircled{A}} &= 1 + \frac{\textcircled{B} - \textcircled{A}}{\textcircled{A}} = 1 - \frac{q^{**}}{\prod_{j=1}^{s+1} M_j \cdot (6n^2)^{s+1} \cdot \textcircled{A}} \text{ with } q^{**} \in \mathbb{N} \\
&\leq 1 - \frac{1}{\prod_{j=1}^{s+1} M_j \cdot (6n^2)^{s+1} \cdot \textcircled{A}} \leq 1 - \frac{1}{\prod_{j=1}^{s+1} M_j \cdot (6n^2)^{s+1} \cdot \sum_{k=0}^{n-1} \prod_{j=1}^{s+1} \left(1 + \frac{\gamma_j}{6}\right)} \\
&= 1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} M_j \left(1 + \frac{\gamma_j}{6}\right)}.
\end{aligned}$$

So in the end, if we have for fixed n that

$$1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} M_j \left(1 + \frac{\gamma_j}{6}\right)} \leq \prod_{j=s+2}^{d} \frac{\left(1 - \frac{\gamma_j}{12}\right)}{\left(1 + \frac{\gamma_j}{6}\right)}, \tag{$*$}$$

then we immediately obtain that

$$\frac{\textcircled{B}}{\textcircled{A}} \leq 1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} M_j \left(1 + \frac{\gamma_j}{6}\right)} \leq \prod_{j=s+2}^{d} \frac{\left(1 - \frac{\gamma_j}{12}\right)}{\left(1 + \frac{\gamma_j}{6}\right)}.$$

Thus, our selection of z_s^B and z_{s+1}^B at the steps s and $s+1$ leads to the best possible branch vector as the above inequality implies that the worst choice of $z_{s+2}, \ldots, z_d$ for the optimal branch is better than the optimal choice of $z_{s+2}, \ldots, z_d$ in any alternative branch.

In particular, if we have weights of the form $\gamma_j = \frac{1}{M_j}$ with $M_j \in \mathbb{N}$ then our condition simplifies to

$$1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} \left(M_j + \frac{1}{6}\right)} \leq \prod_{j=s+2}^{d} \frac{\left(1 - \frac{1}{12M_j}\right)}{\left(1 + \frac{1}{6M_j}\right)}. \tag{$**$}$$

If for fixed n, we have extremely decaying weights $\gamma = \{\gamma_j\}_{j=1}^d$ such that our condition ($**$) is satisfied at each step $1 < s < d$, then an algorithm can deliver the best CBC branch by choosing the branch for which $e_{n,d}(z_1, \ldots, z_s^B, z_{s+1}^B)$ is the smallest amongst all branches for which branching occurred in step s. Moreover, if we can determine at which step s branching occurs then it suffices to verify that the condition ($**$) holds at step s in order to select the best branch. The above condition is of a priori nature, i.e. can be checked beforehand, further it enables us to design an algorithm that decides in step $s+1$ which branch has to be further considered.

Example: To receive an impression of how fast the weight sequence $\gamma = \{\gamma_j\}_{j=1}^d$ has to decay in order to satisfy condition (**), we consider weights of the form $\gamma_j = \frac{1}{M_j}$ where $M_j = q^j$ with $q \in \mathbb{N}$. Since we only proved that branching occurs in step two so far, we consider the condition for $s = 2$ and $d = 10$ and further $n = 101$. Calculating the terms on both sides of (**) in high-precision floating-point arithmetic (VPA-function in MATLAB), we see that for $q = 10000$ we have

$$1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} \left(M_j + \frac{1}{6}\right)} = 0.999999999999999999999999999999999999999999\mathbf{56819}$$

$$\leq \prod_{j=s+2}^{d} \frac{\left(1 - \frac{1}{12M_j}\right)}{\left(1 + \frac{1}{6M_j}\right)} = 0.999999999999999999999999999999999999999999\mathbf{75000}$$

and so the condition (**) is fulfilled. The magnitude of q shows that the weights γ_j have to decrease rapidly in order to satisfy the condition. Such fast decaying weight sequences are rather unrealistic, but the above observations form a theoretical basis for the occurrence of branching in the component-by-component algorithm. Further, we note that the above condition is the result of some rather crude estimations and therefore the above results may also hold for weight sequences which do not decrease as drastically as in our example.

In addition, we consider a weight sequence of the form $\gamma = \{\gamma_j\}_{j=1}^d$ where γ_j is given as

$$\gamma_j = \begin{cases} \frac{1}{q_1^j} & \text{for } j = 1, \ldots, s+1\,, \\ \frac{1}{q_2^j} & \text{for } j = s+2, \ldots, d\,. \end{cases}$$

Then, we obtain for $q_1 = 2$, $q_2 = 60$ and as before $s = 2$, $d = 10$ and $n = 101$ that

$$1 - \frac{1}{n \cdot (6n^2)^{s+1} \cdot \prod_{j=1}^{s+1} \left(M_j + \frac{1}{6}\right)} = 0.99999999999999999\mathbf{41431}$$

$$\prod_{j=s+2}^{d} \frac{\left(1 - \frac{1}{12M_j}\right)}{\left(1 + \frac{1}{6M_j}\right)} = 0.99999999999999999\mathbf{58655}\,,$$

i.e. the condition (**) is again satisfied. Using the branch algorithm in this situation, we obtain two different branch vectors $z^1, z^2 \in Z_m^d$. For these two generating vectors the worst-case error up to dimension $s+1$ gives

$$e^2_{n,s+1}(z^1_1, \ldots, z^1_{s+1}) = 2.32907979 \cdot 10^{-5} \text{ and } e^2_{n,s+1}(z^2_1, \ldots, z^2_{s+1}) = 2.33124562 \cdot 10^{-5}\,.$$

Since our condition is fulfilled, we expect that also $e^2_{n,d}(z^1) < e^2_{n,d}(z^2)$. Indeed, we find that

$$e^2_{n,d}(z^1) = 2.32908034 \cdot 10^{-5} < e^2_{n,d}(z^2) = 2.33124613 \cdot 10^{-5}\,.$$

Hence, once we have verified that condition (**) holds and $e^2_{n,s+1}(z^1_1,\dots,z^1_{s+1}) < e^2_{n,s+1}(z^2_1,\dots,z^2_{s+1})$, we can neglect z^2 and only need to compute z^1 until dimension d.

This chapter shows that one can overcome the difficulties caused by the occurrence of branching in the CBC algorithm which we observed in our numerical experiments. Nevertheless, branching does not help us to implement an algorithm which provides generating vectors with smaller worst-case errors than the component-by-component construction. Thus, we will consider a different type of algorithm in the subsequent chapter.

7 The successive coordinate search algorithm

7.1 The formulation of the successive coordinate search algorithm

As we have seen in the previous chapters, a standard optimization approach does not work in our problem setting. Since the component-by-component construction can provide generating vectors z with acceptable worst-case errors $e_{n,d}(z)$, we will introduce an algorithm of similar nature as the CBC algorithm.

One advantage of the component-by-component construction is that the algorithm is extensible in the dimension d, i.e. if the current dimension d_1 increases to $d_2 > d_1$, the algorithm does not need to restart completely but can start with the CBC vector of dimension d_1. However, in most practical applications the dimension d is predefined, thus it is reasonable to assume that the dimension d is invariable. Under this assumption we can think of a generalized type of algorithm to find a good generating vector z for a rank-one lattice rule. The pseudocode of this so-called successive coordinate search algorithm (SCS algorithm) is provided below.

Algorithm 4 Successive coordinate search algorithm (SCS)

Input: $z^0 \in Z_n^d$
Output: $z \in Z_n^d$

for $i = 1$ **to** d **do**
  for all $z_i \in Z_n$ **do**
    $$e^2_{n,d}(z_1, \dots, z_{i-1}, z_i, z^0_{i+1}, \dots, z^0_d) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega\left(\left\{\frac{k \cdot z_j}{n}\right\}\right)\right)$$
  end for
  $z_i = \underset{z \in Z_n}{\operatorname{argmin}}\ e^2_{n,d}(z_1, \dots, z_{i-1}, z_i, z^0_{i+1}, \dots, z^0_d)$
end for

Instead of increasing the dimension in every step of the algorithm, we keep d fixed during all calculations. Based on a starting vector $z^0 \in Z_n^d$, the algorithm successively selects the coordinate $z_i \in Z_n$ which minimizes the squared worst-case error $e^2_{n,d}(z_1, \dots, z_{i-1}, z_i, z^0_{i+1}, \dots, z^0_d)$ while keeping all other coordinates of z fixed. Thus, in the process of the SCS algorithm every coordinate of the starting vector z^0 is altered in each step of the algorithm. We see that our construction is very similar to the component-by-component construction, with the only difference being that we select an initial vector z^0 as input for our algorithm. In fact, we can prove that the successive coordinate search algorithm is only a generalized version of the CBC algorithm as the following theorem shows.

Theorem 7.1
The component-by-component algorithm is equivalent to the successive coordinate search algorithm with starting vector $z^0 = (0, \dots, 0)$, i.e. the CBC algorithm and the SCS algorithm with starting vector $z^0 = (0, \dots, 0)$ both yield the same generating vector as outcome.

Proof. We denote by 0^r the r-dimensional zero vector, where $1 \le r \le d$. For an arbitrary $z \in Z_n^s$ consider the squared worst-case error $e^2_{n,d}(\tilde{z}) = e^2_{n,d}(z, 0^{d-s})$:

$$
\begin{aligned}
e^2_{n,d}(\tilde{z}) &= e^2_{n,d}(z, 0^{d-s}) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot \tilde{z}_j}{n} \right\} \right) \right) \\
&= -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{s} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) \cdot \prod_{j=s+1}^{d} (\beta_j + \gamma_j \cdot \omega(0)) \\
&= -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \prod_{j=s+1}^{d} (\beta_j + \gamma_j \cdot \omega(0)) \cdot \sum_{k=0}^{n-1} \prod_{j=1}^{s} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) \\
&= -\prod_{j=1}^{d} \beta_j + \frac{C_s}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{s} \left(\beta_j + \gamma_j \cdot \omega \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) \\
&= -\prod_{j=1}^{d} \beta_j + C_s \left(e^2_{n,s}(z) + \prod_{j=1}^{s} \beta_j \right) ,
\end{aligned}
$$

where $C_s := \prod_{j=s+1}^{d} (\beta_j + \gamma_j \cdot \omega(0))$. Here, we assume without loss of generality that we have positive weights $\gamma_j > 0$ and constants $\beta_j > 0$. Additionally, we note that due to the non-negativity of the squared worst-case error $e^2_{n,d}$ the function ω is such that $\omega(0) > 0$. This implies that the constants C_s are positive for all $s = 1, \ldots, d$. Now consider each step in both algorithms:

In step s of the SCS algorithm with initial vector $z^0 = (0, \ldots, 0)$ and $1 \leq s \leq d$, we search for the $\bar{z} \in Z_n$ that minimizes $e^2_{n,d}(z^*_1, \ldots, z^*_{s-1}, \bar{z}, 0^{d-s})$, where $z^*_1, \ldots, z^*_{s-1}$ have been determined in the previous steps of the algorithm. Further, in step s of the component-by-component algorithm we search $\bar{z} \in Z_n$ such that $e^2_{n,s}(z_1, \ldots, z_{s-1}, \bar{z})$ is minimized. Now, if we start off with the first step of both algorithms ($s = 1$), then we minimize $e^2_{n,1}(\bar{z})$ in the CBC algorithm and

$$
e^2_{n,d}(\bar{z}, 0^{d-1}) = -\prod_{j=1}^{d} \beta_j + C_1 \left(e^2_{n,1}(\bar{z}) + \beta_1 \right) ,
$$

in the SCS algorithm, where $\bar{z} \in Z_n$. Since almost all expressions from the last equation are constant, minimizing $e^2_{n,d}(\bar{z}, 0^{d-1})$ is equivalent to minimizing $e^2_{n,1}(\bar{z})$. Thus, the first coordinate of the resulting generating vectors of both algorithms is equal, where we assume w.l.o.g. that if multiple minimizers occur both algorithms select the same. Inductively, we see that in each step of the SCS algorithm we minimize

$$
e^2_{n,d}(z^*_1, \ldots, z^*_{s-1}, \bar{z}, 0^{d-s}) = -\prod_{j=1}^{d} \beta_j + C_s \left(e^2_{n,s}(z^*_1, \ldots, z^*_{s-1}, \bar{z}) + \prod_{j=1}^{s} \beta_j \right) ,
$$

which is equivalent to minimizing $e^2_{n,s}(z^*_1, \ldots, z^*_{s-1}, \bar{z})$ which is also minimized in the CBC algorithm. Therefore, we obtain that both algorithms yield the same generating vector z. ■

Our hope regarding the SCS algorithm is to obtain generating vectors with smaller error values as in the CBC algorithm, provided we choose a suitable initial vector z^0. The formulation of the algorithm suggests that performance of the successive coordinate search construction strongly depends on the starting vector z^0 which we select beforehand. In order to assess the performance of the successive coordinate search algorithm, we conduct some numerical experiments in the same setting as for the CBC algorithm. Since we do not know how to choose the initial vectors for the SCS algorithm, we randomly choose $p = 100$ starting vectors $z^0 \in Z_m^d$ with uniformly distributed coordinates, apply the SCS algorithm to them and then calculate the average value of the corresponding worst-case errors. Moreover, we will display the smallest worst-case error of all generating vectors obtained by the SCS method. We have similar notations as before:

$$z_S = \text{SCS vector}, z^* = \text{ CBC vector}, z_{\text{ex}} = \text{ vector obtained by exhaustive search}$$

Moreover, the squared worst-case error is given as

$$e_{n,d}^2(z_1, \ldots, z_d) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot B_2 \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) .$$

Numerical Experiments:

Table 9: $d = 4, \beta_j = 1, \gamma_j = 1/j^2$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$
101	6.6252e-03	6.5650e-03	6.6448e-03	6.5650e-03
127	5.3293e-03	5.2944e-03	5.3093e-03	5.2944e-03
139	4.9044e-03	4.8718e-03	4.8921e-03	4.8718e-03
151	4.5160e-03	4.4795e-03	4.4795e-03	4.4795e-03
181	3.7652e-03	3.7322e-03	3.7425e-03	3.7322e-03
199	3.4725e-03	3.4274e-03	3.4485e-03	3.4274e-03

Table 10: $d = 4, \beta_j = 1, \gamma_j = 1/j^3$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$
101	5.1181e-03	5.1093e-03	5.1093e-03	5.1093e-03
127	4.1236e-03	4.1145e-03	4.1161e-03	4.1145e-03
139	3.7666e-03	3.7606e-03	3.7623e-03	3.7606e-03
151	3.4575e-03	3.4475e-03	3.4475e-03	3.4475e-03
181	2.8864e-03	2.8750e-03	2.8784e-03	2.8750e-03
199	2.6397e-03	2.6310e-03	2.6310e-03	2.6310e-03

Table 11: $d = 4, \beta_j = 1, \gamma_j = (0.9)^j$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$
101	1.7016e-02	1.6807e-02	1.7242e-02	1.6798e-02
127	1.3698e-02	1.3500e-02	1.3681e-02	1.3495e-02
139	1.2790e-02	1.2640e-02	1.2850e-02	1.2633e-02
151	1.2006e-02	1.1741e-02	1.1991e-02	1.1738e-02
181	1.0043e-02	9.7840e-03	1.0120e-02	9.7840e-03
199	9.2766e-03	9.0730e-03	9.2822e-03	9.0730e-03

Table 12: $d = 4, \beta_j = 1, \gamma_j = (0.5)^j$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$
101	5.2711e-03	5.2183e-03	5.2692e-03	5.2183e-03
127	4.2086e-03	4.1760e-03	4.1992e-03	4.1760e-03
139	3.8814e-03	3.8578e-03	3.8641e-03	3.8578e-03
151	3.5914e-03	3.5484e-03	3.5484e-03	3.5484e-03
181	2.9836e-03	2.9501e-03	2.9586e-03	2.9501e-03
199	2.7438e-03	2.7106e-03	2.7169e-03	2.7106e-03

Table 13: $d = 4, \beta_j = 1, \gamma_j = (0.1)^j$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$	$e_{n,d}(z_{\text{ex}})$
101	1.3615e-03	1.3614e-03	1.3615e-03	1.3614e-03
127	1.0842e-03	1.0842e-03	1.0842e-03	1.0842e-03
139	9.9033e-04	9.9031e-04	9.9031e-04	9.9031e-04
151	9.1136e-04	9.1135e-04	9.1136e-04	9.1135e-04
181	7.6024e-04	7.6017e-04	7.6017e-04	7.6017e-04
199	6.9167e-04	6.9166e-04	6.9166e-04	6.9166e-04

Analysis of the results: Even though the averaged worst-case error is often larger than the error value of the CBC vector, we see that the minimal error values amongst the $p = 100$ different generating vectors obtained by the SCS algorithm almost always attain the absolute minimum in the set Z_n^d of all possible generating vectors. This means the SCS algorithm delivers generating vectors z with the same worst-case error $e_{n,d}(z)$ as for the best generating vector obtained by exhaustive search. In particular, the SCS algorithm outperforms the component-by-component construction for slowly decaying weight sequences, e.g. for $\gamma_j = (0.9)^j$.

Remark: The occurrence of multiple minimizers, which we examined in the last chapter, plays also a role for the SCS construction. Similarly, it is possible that we obtain various minimizers in a particular minimization step of the algorithm, which result in different SCS vectors that also have different error values. As before, we can apply condition (**) to find the best of these SCS vectors.

These very promising results encourage us to pursue the analysis of the successive coordinate search algorithm. As for the component-by-component construction, one can implement a fast version of the successive coordinate search algorithm by using a fast Fourier transformation to compute the matrix-vector-product in each step of the SCS algorithm. In order to assess the computational cost of the SCS algorithm, we can again rewrite the error term

$$e^2_{n,d}(z_1,\ldots,z_{i-1},\bar{z},z^0_{i+1},\ldots,z^0_d) = -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(1+\gamma_j\cdot\omega\left(\left\{\frac{k\cdot\tilde{z}_j}{n}\right\}\right)\right)$$

which has to be calculated in each step of the algorithm as

$$e^2_{n,d}(z_1,\ldots,z_{i-1},\bar{z},z^0_{i+1},\ldots,z^0_d) = -1+\frac{1}{n}\sum_{k=0}^{n-1}\prod_{\substack{j=1\\ j\neq i}}^{d}\left(1+\gamma_j\cdot\omega\left(\left\{\frac{k\cdot\tilde{z}_j}{n}\right\}\right)\right)\left(1+\gamma_i\cdot\omega\left(\left\{\frac{k\cdot z}{n}\right\}\right)\right)$$

where $\bar{z}\in Z_n$ and we define $\tilde{z}_j$ as follows

$$\tilde{z}_j = \begin{cases} z_j & \text{for } j<i\,,\\ \bar{z} & \text{for } j=i\,,\\ z^0_j & \text{for } j>i\,. \end{cases}$$

Furthermore, we define for every $i=1,\ldots,d$ the n-dimensional vector p_i with components given as

$$p_i(k) := \prod_{\substack{j=1\\ j\neq i}}^{d}\left(1+\gamma_j\cdot\omega\left(\left\{\frac{k\cdot\tilde{z}_j}{n}\right\}\right)\right)\,,$$

where the components $\tilde{z}_j$ belong to the current vector $z\in Z_n^d$ which is updated in each step of the algorithm. In contrast to our analysis of the component-by-component construction, p_i has to be updated in every step of the algorithm according to the following iteration

$$p_{i+1}(k) = p_i(k)\cdot\frac{1+\gamma_i\cdot\omega\left(\left\{\frac{k\cdot\tilde{z}_i}{n}\right\}\right)}{1+\gamma_{i+1}\cdot\omega\left(\left\{\frac{k\cdot\tilde{z}_{i+1}}{n}\right\}\right)}\,.$$

Hence, we can isolate the major part of the calculation by defining (for all $\bar{z}\in Z_n$)

$$v_i(\bar{z}) := \sum_{k=0}^{n-1} p_i(k)\cdot\omega\left(\left\{\frac{k\cdot\bar{z}}{n}\right\}\right) = (\Omega_n\cdot p_i)(\bar{z})\,,$$

where the matrix Ω_n is again given as

$$\Omega_n := \left[\omega\left(\left\{\frac{k\cdot z}{n}\right\}\right)\right]_{\substack{z=1,\ldots,n-1\\ k=0,\ldots,n-1}}\,.$$

Then we have that $e^2_{n,d}(z_1,\dots,z_{i-1},\bar{z},z^0_{i+1},\dots,z^0_d) = -1 + \frac{1}{n}\sum_{k=0}^{n-1} p_i(k) + \frac{\gamma_i}{n}\cdot v_i(\bar{z})$. Using fast Fourier transformation, we can reduce the complexity of the matrix-vector product to $\mathcal{O}(n\,\log(n))$ and so the time complexity to calculate the squared worst-case error $e^2_{n,d}(z_1,\dots,z_{i-1},\bar{z},z^0_{i+1},\dots,z^0_d)$ for all $\bar{z}\in Z_n$ in each step of the algorithm is $\mathcal{O}(n\,\log(n))$. Thus, the resulting algorithm in d dimensions has again time complexity $\mathcal{O}(d\,n\,\log(n))$ and so we are able to compute SCS vectors for large values of d and n in a fast manner. In particular, this fast version of the algorithm allows us to use many different starting vectors $z^0\in Z_m^d$, i.e. use a large number p. The corresponding MATLAB code for such a fast version of the successive coordinate search algorithm can be found below. Note that we assume that the kernel function ω is symmetric around $\frac{1}{2}$ and therefore only select components from the restricted search space Z_m. Moreover, we only allow initial vectors z^0 from the restricted set Z_m^d.

Listing 2: `Fast version of the SCS algorithm`

```
% Author: Adrian Ebert
% Based on the fast CBC algorithm by Dirk Nuyens
%
% function [z, e2] = Fast_SCS(n, z0, omega, gamma, beta)
%
% inputs
%   n           number of quadrature points, integer, scalar
%   z_0         initial vector with components in Z_m, integer, 1 x d_max vector
%   omega       function handle for kernel function \omega
%   gamma       weight sequence, real, 1 x d_max vector
%   beta        constant sequence, real, 1 x d_max vector
%
% outputs
%   z           generating vector of the lattice rule, d_max x 1 vector
%   e2          squared worst-case error per dimension, d_max x 1 vector

function [z, e2] = Fast_SCS(n, z0, omega, gamma, beta)

if ~isprime(n), error('n must be prime'); end;

d_max = length(z0);
z = zeros(d_max, 1);
m = (n-1)/2;     % assume the omega function is symmetric around 1/2

g = generatorp(n);
perm = zeros(m, 1);
perm(1) = 1; for j=1:m-1, perm(j+1) = mod(perm(j)*g, n); end;
perm = min(n - perm, perm);
psi = omega(perm/n);
fft_psi = fft(psi);
q = ones(m, 1);

[~,perminv] = sort(perm); % inverse permutation map
w0 = perminv(z0); % inverse permuation of start vector

for d = 1:d_max
   q = (beta(d) + gamma(d) * psi([w0(d):-1:1 m:-1:w0(d)+1])) .* q;
```

```
end

for d = 1:d_max

    q = q ./ (beta(d) + gamma(d) * psi([w0(d):-1:1 m:-1:w0(d)+1]));
    E2 = ifft(fft_psi .* fft(q));
    E2 = real(E2);
    [~, w] = min(E2);
    z(d) = perm(w);
    q = (beta(d) + gamma(d) * psi([w:-1:1 m:-1:w+1])) .* q;

end
e2 = wce(n, z', omega, gamma, beta);
end
```

7.2 Numerical results and experiments

This fast version of the SCS algorithm enables us to further examine the nature of the algorithm for a larger number n of quadrature points and larger dimensions d. Therefore, we display the results of some numerical experiments with moderately high dimensions of $d = 20$ and $n = 1009$ below. In the experiments, we do again compare the worst-case errors of the CBC vector and the best SCS vector and the average error value of all generated SCS vectors ($p = 1000$). Here, the squared worst-case error takes the following form

$$e^2_{n,d}(z_1, \dots, z_d) = -\prod_{j=1}^{d} \beta_j + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{d} \left(\beta_j + \gamma_j \cdot B_2 \left(\left\{ \frac{k \cdot z_j}{n} \right\} \right) \right) .$$

Remark: For large dimensions d and a large number of quadrature points n it is unfortunately not possible to determine the best generating vector z_{ex} in Z_m^d via an exhaustive search as we did for small dimensions d. The only possibility to assess how good a particular generating vector obtained by the SCS algorithm is, is to compare its worst-case error with the error of the CBC vector.

Numerical Experiments:

Table 14: $d = 20, \beta_j = 1, \gamma_j = (0.95)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$
101	1.6386e-01	1.6213e-01	1.6453e-01
199	1.0838e-01	1.0641e-01	1.0758e-01
307	8.3156e-02	8.1406e-02	8.2472e-02
401	7.0631e-02	6.9335e-02	6.9717e-02
601	5.5116e-02	5.4181e-02	5.4608e-02
809	4.5836e-02	4.4923e-02	4.5167e-02

Table 15: $d = 20, \beta_j = 1, \gamma_j = (0.8)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$
101	3.0985e-02	3.0144e-02	3.0569e-02
199	1.8699e-02	1.7921e-02	1.8194e-02
307	1.3454e-02	1.2993e-02	1.2732e-02
401	1.1014e-02	1.0592e-02	1.0697e-02
601	8.1173e-03	7.7968e-03	7.8423e-03
809	6.5005e-03	6.1438e-03	6.1895e-03

Table 16: $d = 20, \beta_j = 1, \gamma_j = (0.6)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	$\min_{z_S} e_{n,d}(z_S)$	$e_{n,d}(z^*)$
101	8.7869e-03	8.5857e-03	8.5831e-03
199	4.8218e-03	4.6566e-03	4.6684e-03
307	3.2857e-03	3.1510e-03	3.1596e-03
401	2.6001e-03	2.4820e-03	2.4958e-03
601	1.8252e-03	1.7466e-03	1.7495e-03
809	1.3987e-03	1.3120e-03	1.3105e-03

We see that in all cases and for all used weight sequences γ the averaged worst-case error $\overline{e_{n,d}(z_S)}$ is larger than the error of the CBC vector. Nevertheless, we also notice that the best SCS vector in our sample set of cardinality $p = 1000$ mostly possesses a smaller worst-case error than the related CBC vector. However, our experiments showed that the sample size p has to be increased in order to reliably find generating vectors with smaller error values than the component-by-component construction.

In order to assess the quality of the successive coordinate search algorithm, we are in particular interested in the convergence rate of the algorithm. Based on Theorem 5.3 by Kuo, we expect that the worst-case error is bounded as follows

$$e_{n,d}(z) \leq C_d \cdot n^{-\omega} \cdot e_{0,d}\,.$$

Hence, we assume that $e_{n,d}(z) = \mathcal{O}(n^{-\omega})$ and aim to determine the value of ω. As in [2], we can then use two consecutive pairs $(n_1, e_{n_1,d}(z_1))$ and $(n_2, e_{n_2,d}(z_2))$ to determine the exponent of convergence ω via

$$\omega = \log\left(\frac{e_{n_1,d}(z_1)}{e_{n_2,d}(z_2)}\right) \Big/ \log\left(\frac{n_2}{n_1}\right).$$

The following tables display the convergence rates of the CBC construction, the minimal SCS vector and the averaged worst-case error $\overline{e_{n,d}(z_S)}$ of all used SCS vectors, where we used large values of n and d. For the sake of comparability, we choose a similar setting as in [2] and hence use the

weighted Korobov space with parameters $\alpha = 2$ and $\beta_j = 1$ and the weighted anchored Sobolev space with $c_j = 1$ and $\beta_j = 1$.

Table 17: Weighted Korobov spaces: $d = 100, \alpha = 2, \beta_j = 1, \gamma_j = (0.9)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	4.0301e+02		3.9727e+02		4.0211e+02	
2003	2.8596e+02	0.50043	2.8202e+02	0.49974	2.8555e+02	0.49927
4001	2.0226e+02	0.50055	1.9985e+02	0.49789	2.0242e+02	0.49734
8009	1.4297e+02	0.49993	1.4142e+02	0.49833	1.4256e+02	0.50524
16001	1.0113e+02	0.50021	9.9888e+01	0.50241	1.0109e+02	0.49661

Table 18: Weighted Korobov spaces: $d = 100, \alpha = 2, \beta_j = 1, \gamma_j = (0.5)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	2.9265e-02		2.7995e-02		2.8356e-02	
2003	1.7679e-02	0.73507	1.6805e-02	0.74427	1.7013e-02	0.74502
4001	1.0558e-02	0.74498	9.9664e-03	0.75512	9.8750e-03	0.78624
8009	6.2778e-03	0.74910	5.9527e-03	0.74259	5.9293e-03	0.73499
16001	3.7186e-03	0.75664	3.5267e-03	0.75639	3.5558e-03	0.73882

Table 19: Weighted Korobov spaces: $d = 100, \alpha = 2, \beta_j = 1, \gamma_j = (0.1)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	7.6697e-04		7.6515e-04		7.6628e-04	
2003	3.9176e-04	0.97976	3.8947e-04	0.98484	3.8971e-04	0.98609
4001	2.0028e-04	0.96971	1.9935e-04	0.96795	1.9988e-04	0.96500
8009	1.0265e-04	0.96300	1.0236e-04	0.96037	1.0241e-04	0.96349
16001	5.2186e-05	0.97752	5.1816e-05	0.98377	5.1867e-05	0.98303

Table 20: Weighted Korobov spaces: $d = 100, \alpha = 2, \beta_j = 1, \gamma_j = 1/j^2, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	7.8020e-02		7.6395e-02		7.7139e-02	
2003	5.0214e-02	0.64268	4.9262e-02	0.63988	4.9174e-02	0.65663
4001	3.2103e-02	0.64654	3.1519e-02	0.64541	3.1264e-02	0.65456
8009	2.0436e-02	0.65076	2.0015e-02	0.65430	1.9793e-02	0.65869
16001	1.2999e-02	0.65377	1.2713e-02	0.65584	1.2550e-02	0.65836

Table 21: Weighted Sobolev spaces: $d = 100, c_j = 1, \beta_j = 1, \gamma_j = (0.9)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	8.1659e-02		8.0011e-02		7.9322e-02	
		0.65109		0.65326		0.64697
2003	5.2254e-02		5.1123e-02		5.0902e-02	
		0.65166		0.65951		0.66815
4001	3.3289e-02		3.2392e-02		3.2060e-02	
		0.65168		0.64034		0.66828
8009	2.1178e-02		2.0770e-02		2.0162e-02	
		0.65363		0.65759		0.65379
16001	1.3472e-02		1.3176e-02		1.2824e-02	

Table 22: Weighted Sobolev spaces: $d = 100, c_j = 1, \beta_j = 1, \gamma_j = (0.5)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	7.4315e-04		7.0982e-04		7.1755e-04	
		0.92998		0.92175		0.92848
2003	3.9277e-04		3.7728e-04		3.7963e-04	
		0.93024		0.94161		0.94255
4001	2.0635e-04		1.9666e-04		1.9776e-04	
		0.91882		0.91844		0.92768
8009	1.0906e-04		1.0397e-04		1.0388e-04	
		0.92215		0.94059		0.92081
16001	5.7611e-05		5.4222e-05		5.4924e-05	

Table 23: Weighted Sobolev spaces: $d = 100, c_j = 1, \beta_j = 1, \gamma_j = (0.1)^j, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	1.3738e-04		1.3737e-04		1.3737e-04	
		0.99882		0.99895		0.99895
2003	6.9258e-05		6.9247e-05		6.9247e-05	
		0.99763		0.99753		0.99750
4001	3.4729e-05		3.4726e-05		3.4727e-05	
		0.99702		0.99710		0.99704
8009	1.7385e-05		1.7383e-05		1.7384e-05	
		0.99888		0.99893		0.99897
16001	8.7087e-06		8.7071e-06		8.7074e-06	

Table 24: Weighted Sobolev spaces: $d = 100, c_j = 1, \beta_j = 1, \gamma_j = 1/j^2, p = 1000$

n	$\overline{e_{n,d}(z_S)}$	ω_1	$\min_{z_S} e_{n,d}(z_S)$	ω_2	$e_{n,d}(z^*)$	ω_3
1009	1.3529e-03		1.3015e-03		1.2902e-03	
		0.87681		0.88763		0.89220
2003	7.4156e-04		7.0814e-04		6.9978e-04	
		0.87626		0.86704		0.88835
4001	4.0443e-04		3.8867e-04		3.7846e-04	
		0.86583		0.87599		0.89194
8009	2.2175e-04		2.1162e-04		2.0379e-04	
		0.86311		0.85718		0.87428
16001	1.2202e-04		1.1693e-04		1.1128e-04	

Our results show that the convergence rates of the component-by-component construction, the SCS construction and the averaged worst-case error $\overline{e_{n,d}(z_S)}$ are nearly identical. Furthermore, we see that the faster the weights γ_j decline the larger the exponent of convergence ω is. Moreover, our experiments also revealed that we have to use a larger number p of initial vectors z_0 for the successive coordinate search algorithm in order to find generating vectors with smaller error values than the component-by-component construction can provide. In our experiments we searched for good generating vectors z by applying the SCS method to p different randomly distributed starting vectors z_0. Due to that, the time complexity of the applied SCS heuristic is of order $\mathcal{O}(p\,d\,n\,\log(n))$. For large values of p this heuristic is therefore notably slower than the fast version of the component-by-component construction which has time complexity $\mathcal{O}(d\,n\,\log(n))$. Nevertheless, the fast version of the SCS algorithm still assures that the calculation of generating vectors can be executed in reasonable time.

Furthermore, it is also possible to improve existing generating vectors by applying the SCS algorithm to them. An obvious candidate for an initial vector for such a procedure would be the CBC vector, since then the SCS construction promises that the resulting vector has a smaller worst-case error than the initial CBC vector. However, our experiments showed that in most cases the CBC vector is a local minimum with respect to the successive coordinate search algorithm, i.e. applying the SCS algorithm to the CBC vector yields no improvement. Nevertheless there were cases in which we could achieve a reduction of the worst-case error by using the above procedure.

Example: For a dimension of $d = 50$ and $n = 2003$ we obtain for the weighted Korobov space with smoothing parameter $\alpha = 2$, $\beta_j = 1$ and weights of the form $\gamma_j = (0.99)^j$ that

$$e_{n,d}(z_{\mathrm{CBC}}) = 1.2857 \cdot 10^{12}\,,$$

where z_{CBC} is the generating vector obtained by the CBC algorithm in the respective function space. Further, if we apply the SCS algorithm to z_{CBC}, we obtain the generating vector z_{SCS} with corresponding worst-case error

$$e_{n,d}(z_{\mathrm{SCS}}) = 1.2440 \cdot 10^{12}\,.$$

Thus, the above heuristic yields a notable reduction of the worst-case error of the CBC vector.

The observations and experiments in the previous sections showed that it is possible to use the successive coordinate search algorithm to construct good generating vectors for rank-one lattice rules. They also confirmed that methods based on the SCS construction can provide generating vectors with smaller worst-case errors than the CBC vector. However, the computational cost of those methods is several times higher than for the component-by-component construction, which excels due to its remarkable speed. Additionally, the improvements due to SCS were mostly only marginal. For those reasons, the SCS algorithm should be regarded as a generalization of the existing component-by-component construction rather than a completely new algorithm. Above, we have already shown numerically that our algorithm has a similar convergence rate as the CBC algorithm, it remains to prove theoretically that the successive coordinate search algorithm achieves the same optimal rates of convergence as the component-by-component construction (compare to [2]).

8 Conclusion

In the course of this thesis, we have intensively studied the component-by-component construction which is used to find good generating vectors z for rank-one lattice rules in weighted reproducing kernel Hilbert spaces. We have seen that due to the ill-behaved nature of the squared worst-case error $e^2_{n,d}(z)$ given as

$$e^2_{n,d}(z_1,\dots,z_d) = -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{d}\left(1+\gamma_j\cdot\omega\left(\left\{\frac{k\cdot z_j}{n}\right\}\right)\right),$$

it is not possible to approach the problem to find a generating vector $z \in Z_n^d$ such that $e^2_{n,d}(z)$ is minimized by means of standard optimization methods.

Moreover, we have analysed the occurrence of multiple minimizers in the steps of the component-by-component algorithm which resulted in the fact that there exist more than one CBC vector for a particular setting. As we have seen, these related branch vectors often had notably different worst-case errors, so that the choice of the best of these vectors became important. The derived conditions in chapter 6 allowed us to select the best branch vector without having to compute each branch until dimension d, provided the weight sequence at hand satisfies condition (**).

At the end, we provided a generalized version of the component-by-component construction, the so-called successive coordinate search algorithm. This algorithm finds good generating vectors for rank-one lattice rules, especially when the number of quadrature points n and the respective dimension d are moderately high. Nevertheless, we had to do draw a large number of initial vectors from a random distribution and then apply the SCS algorithm to them, which resulted in increased computational costs. However, we could implement a fast version of the SCS method (based on the fast CBC algorithm by Dirk Nuyens) that made these increased costs less important. Further, we numerically proved that our algorithm exhibits similar convergence rates as the original CBC construction.

In summary, it can be said that the CBC construction works remarkably well, especially considering that in the context of the SCS algorithm the CBC algorithm is equivalent to the SCS algorithm with starting point $(0,\dots,0)$, which is the worst possible starting point as the worst-case error attains its maximum there (for $\omega(\cdot) = B_2(\cdot)$). Furthermore, our experiments showed that the expected improvements of the component-by-component construction can only be marginal since the CBC vector and the best possible generating vector usually have error values of the same magnitude. Nevertheless, our analysis has given us valuable insight in the nature of the component-by-component algorithm and has furthermore generalized the existing construction. Additionally, it is possible to use the successive coordinate search algorithm to improve existing lattice rules by using the corresponding generating vector as an initial vector for the SCS algorithm.

9 Appendix

In this section we provide additional MATLAB programs and functions which we used for our numerical experiments.

9.1 Function err

This function calculates the standard squared worst-case error of a specific generating vector z for the weighted unanchored Sobolev space. The corresponding function space which determines the form of the worst-case error expression can be specified via the input parameters n and the weight sequence $\gamma = \{\gamma_j\}_{j=1}^{d}$.

Listing 3: `Function to calculate the standard squared worst-case error`

```
% Author: Adrian Ebert
%
% standard squared worst-case error functional
%
% function [E] = err(n,gamma,z)
%
% inputs
%   n           number of quadrature points, integer, scalar
%   gamma       weight sequence, real, 1 x d vector
%   z           generating vector, integer, 1 x d vector
% outputs
%   E           squared worst-case error of z

function [E] = err(n,gamma,z)

s_max = length(z);
C = ones(n,1);

A = (0:n-1)'*z;
A = mod(A,n)/n;
A = A.^2 - A + (1/6)*ones(n,s_max);

for i=1:s_max

    A(:,i)=ones(n,1)+gamma(i)*A(:,i);
    C=C.*A(:,i);

end

E = -1 + sum(C)/n;

end
```

9.2 Function wce

This function calculates the worst-case error of a specific generating vector z. The corresponding function space which determines the form of the worst-case error expression can be specified via the different input parameters.

Listing 4: `Function to calculate the squared worst-case error`

```
% Author: Adrian Ebert
%
% squared worst-case error functional
%
% function [E] = wce(n, z, omega, gamma, beta)
%
% inputs
%   n           number of quadrature points, integer, scalar
%   z           generating vector, integer, 1 x d vector
%   omega       function handle for kernel function \omega
%   gamma       weight sequence, real, 1 x d_max vector
%   beta        constant sequence, real, 1 x d_max vector
% outputs
%   E           squared worst-case error of z

function [E] = wce(n, z, omega, gamma, beta)

d_max = length(z);
C = ones(n,1);

A = (0:n-1)'*z;
A = mod(A,n)/n;
A = omega(A);

for i=1:d_max

    A(:,i) = beta(i)*ones(n,1) + gamma(i)*A(:,i);
    C = C.*A(:,i);

end

E = -prod(beta) + sum(C)/n;

end
```

9.3 Function CBC_fast

This function executes the standard component-by-component construction which searches for a good generating vector $z \in Z_n^d$. Again, the different input parameters determine the respective reproducing kernel Hilbert space and thereby the worst-case error expression which is minimized in each step of the algorithm. Moreover, the function returns the squared worst-case error $e_{n,j}^2(z)$ for each step, i.e. $j = 1, \ldots, d$.

Listing 5: `Standard component-by-component algorithm`

```
% Author: Adrian Ebert
%
% function [z,e2] = CBC_fast(n, d_max, omega, gamma, beta)
%
% inputs
%   n           number of quadrature points, integer, scalar
%   d_max       dimension, integer, scalar
%   omega       function handle for kernel function \omega
%   gamma       weight sequence, real, 1 x d_max vector
%   beta        constant squence, real, 1 x d_max vector
%
% outputs
%   z           generating vector for the lattice rule
%   e2          squared worst-case error per dimension, d_max x 1 vector

function [z,e2] = CBC_fast(n, d_max, omega, gamma, beta)

p = ones(1,n); E2 = zeros(n,1);
e2 = zeros(d_max,1); z = ones(d_max,1);

M = (1:n-1)'*(0:n-1);
M = omega(mod(M,n)/n);

for d=1:d_max

    [e2(d),z(d)] = min(E2);

    if (d == d_max), break; end;

    p = p.*(beta(d)*ones(1,n) + gamma(d)*M(z(d),:));
    E2 = (-prod(beta(1:d)) + sum(p)/n)*ones(n-1,1) + (gamma(d+1)/n)*(M*p');

end
end
```

There were many other programs and scripts which we used to conduct the numerical experiments and to create the different images in this thesis. We have included the code of the most important algorithms and functions in the work and this appendix, additionally this thesis is handed in with an attached USB flash drive which contains all programs that were used.

References

[1] J. Dick, F.Y. Kuo, I.H. Sloan, Acta Numerica: High dimensional integration - the Quasi-Monte Carlo way, Cambridge University Press 2014.

[2] F.Y. Kuo, Component-by-component constructions achieve the optimal rate of convergence for multivariate integration in weighted Korobov and Sobolev spaces, J. Complexity 19 (2003) 301–320.

[3] G. Leobacher, F. Pillichshammer, Introduction to Quasi-Monte Carlo integration and applications, Springer 2014.

[4] E. Novak, H. Woźniakowski, Tractability of Multivariate Problems, Volume III: Standard Information for Operators, EMS, Zurich (2012)

[5] D. Nuyens, R. Cools, Fast component-by-component construction of rank-1 lattice rules with a non-prime number of points, J.Complexity 22 (2006) 4-28.

[6] S. Paskov, New methodologies for valuing derivatives, Technical Report, Comb. Sci. Dept., Columbia University, October, 1994

[7] S. Paskov, J. Traub, Faster Valuation of Financial Derivatives, The Journal of Portfolio Management, Vol. 22, No. 1, 113-120, Fall 1995.

[8] I.H. Sloan, F.Y. Kuo, S. Joe, On the step-by-step construction of quasi-Monte Carlo integration rules that achieve strong tractability error bounds in weighted Sobolev spaces, Math. Comput. 71 (2002) 1609–1640.

[9] I.H. Sloan, F.Y. Kuo, S. Joe, Constructing randomly shifted lattice rules in weighted Sobolev spaces, SIAM J. Numer. Anal. 40 (2002) 1650–1665.

[10] I.H. Sloan, H. Woźniakowski, When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?, J.Complexity 14 (1998) 1-33.

[11] I.H. Sloan, H. Woźniakowski, Tractability of multivariate integration for weighted Korobov classes, J.Complexity 17 (2001) 697-721.

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe und ich zum ersten Mal eine Masterarbeit in diesem Studiengang einreiche.

Berlin, den 2. November 2015 .....